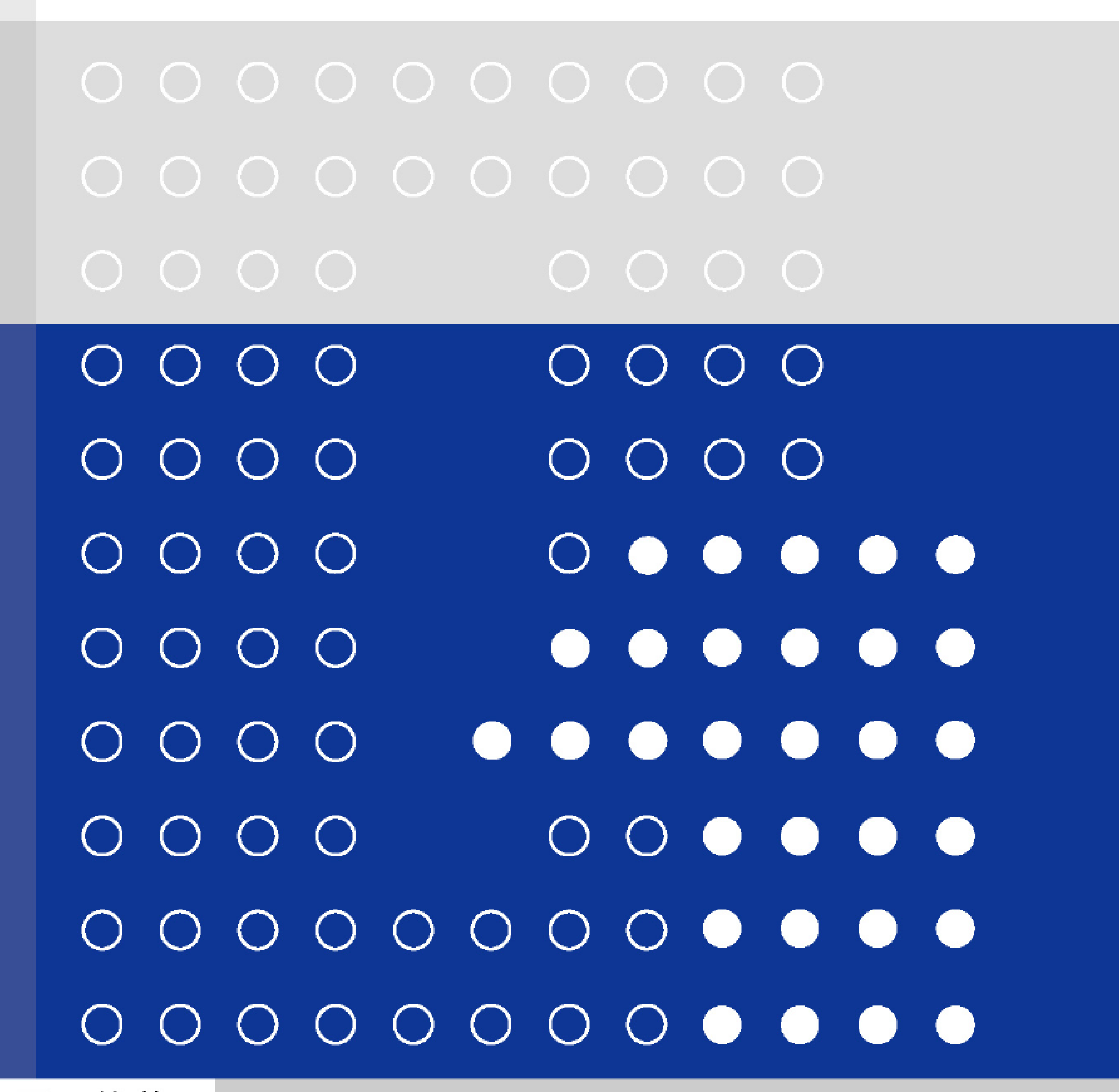
计算机系列教材

Linux操作系统管理与网络服务教程



北京盛浩博远教育科技有限公司　编著

清华大学出版社

计算机系列教材

Linux 操作系统管理与网络服务教程

北京盛浩博远教育科技有限公司　编著

清华大学出版社
北　京

内 容 简 介

本书带领读者进入 Linux 世界，以循序渐进的原则来引导读者学习和掌握 Linux 的使用。本书覆盖了 Linux 从内核到应用的全部核心知识点，使得本书具有完整的 Linux 知识体系。本书包括 4 篇，第 1 篇“Linux 操作系统基础”将引导读者建立 Linux 操作系统，认识 Linux 操作系统的基本使用环境，熟悉 Linux 操作系统的基本操作等；第 2 篇“Linux 操作系统的基本管理”将学习重点转移到操作系统管理操作上，内容包括用户账户管理、文件系统管理、磁盘管理以及系统资源管理等；第 3 篇“Shell 基础”对 Shell 脚本程序设计做了概括性的介绍；第 4 篇“网络服务基础”就常见的 Linux 网络服务器的搭建与基本配置展开讨论。书中结合了大量实践操作案例并辅以操作过程图示。本书作者在编写时参考了 LPI (Linux Professional Institute)认证考试大纲的要求，涵盖了 LPI 101 与 LPI 102 考试大纲要求的知识点。

本书既可作为高等院校 Linux 相关课程的专业教材，也可作为读者参加 LPIC Level 1 认证培训及考试复习的核心学习材料，还可作为专业人士的常用参考书。

图书在版编目 CIP 数据

Linux 操作系统管理与网络服务教程/北京盛浩博远教育科技有限公司编著. —北京：清华大学出版社，2012.4
(计算机系列教材)
ISBN 978-7-302-27601-2

Ⅰ.①L…　Ⅱ.①北…　Ⅲ.①Linux 操作系统—教材　Ⅳ.①TP316.89

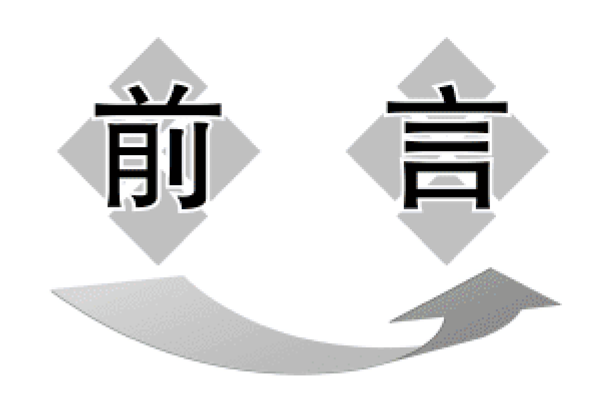
中国版本图书馆 CIP 数据核字(2011)第 272498 号

责任编辑： 龙启铭　战晓雷
封面设计： 傅瑞学
责任校对： 时翠兰
责任印制： 何　芊

出版发行： 清华大学出版社
网　址： http://www.tup.com.cn，http://www.wqbook.com
地　址： 北京清华大学学研大厦 A 座　　**邮　编：** 100084
社 总 机： 010-62770175　　**邮　购：** 010-62786544
投稿与读者服务： 010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈： 010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者： 北京市人民文学印刷厂
装 订 者： 三河市李旗庄少明印装厂
经　销： 全国新华书店
开　本： 185mm×260mm　　**印　张：** 24.75　　**字　数：** 564 千字
版　次： 2012 年 4 月第 1 版　　**印　次：** 2012 年 4 月第 1 次印刷
印　数： 1～3000
定　价： 39.00 元

产品编号：044262-01

前言



Linux 作为 GPL（General Public License）协议下的操作系统，基于其源代码开放的特性，为其良性发展打下了坚实的基础。得益于开源的本质和稳定的性能，Linux 操作系统在当今经济社会各个领域的应用越来越广泛。使用者不仅可以直观地获取该操作系统的实现机制，而且可以根据自身的需要来修改完善这个操作系统，使其最大化地适应用户的需要。这实际是为应用创造了一个平台，而且是一个可以保证性能的高效的、精练的平台。

本书的目的是带领读者进入 Linux 世界，体现了循序渐进的原则。跟随本书的进度，相信读者能从中对 Linux 操作系统得窥一斑，同时通过学习和实践逐渐掌握 Linux 的使用能力。

本书系统全面地覆盖了 Linux 从内核到应用的全部核心知识点，具有完整的 Linux 知识体系。书中大量结合实践操作案例并辅以操作过程的图示，具有良好的实用价值。本书作者在编写时参考了 LPI (Linux Professional Institute) 认证的考试大纲的要求，涵盖了 LPI 101 与 LPI 102 考试大纲要求的知识点。本书既可作为高等院校 Linux 相关课程的专业教材，也可成为读者参加 LPIC Level 1 认证培训及考试复习的核心学习材料，还可作为有经验的专业人士的案头参考资料。

本书包括 4 篇，第 1 篇“Linux 操作系统基础”将引导读者建立 Linux 操作系统，认识 Linux 操作系统的基本使用环境，熟悉 Linux 操作系统中的基本操作，以便于更好、更快地认识与熟悉这个操作系统；第 2 篇“Linux 操作系统的基本管理”将学习重点转移到操作系统管理操作上，内容包括用户账户管理、文件系统管理、磁盘管理以及系统资源管理等一系列题目，熟练地掌握这部分内容是成为合格的系统管理员的必备条件；第 3 篇“Shell 基础”对 Shell 脚本程序设计做了概括性的介绍，力求使读者尽快认识 Shell 及其在系统管理中的作用；第 4 篇“网络服务基础”就常见的 Linux 网络服务器的搭建与基本配置展开了讨论，其目的是为让读者对 Linux 网络服务器有一个基本的认识，为继续学习本书的高级版奠定良好的基础。全书的章节安排如下。

第 1 篇“Linux 操作系统基础”包括以下 4 章:

第 1 章“Linux 概述”介绍 Linux 的产生、发展与发行版等相关背景知识。

第 2 章“Linux 操作系统的部署”介绍 Linux 操作系统中硬盘与分区的表示方法，以及 Linux 操作系统的安装方法。

第 3 章“Linux 的基本操作”介绍 Linux 操作系统界面的使用方法和基本的操作命令。

第 4 章“文本编辑工具 vim”介绍 vim 编辑器的模式、功能与使用方法。

第 2 篇“Linux 操作系统的基本管理”包括以下 13 章:

第 5 章“用户账号和组管理”介绍 Linux 操作系统中的用户账号与用户组的配置管理方法。

第 6 章“文件与目录系统”介绍文件系统结构，特别是文件的权限控制以及 FHS 标准的定义。

第 7 章“Shell 基础”介绍 Shell 的种类及 Shell 的基本功能，重点是 Shell 的基本功能的使用。

第 8 章“Shell 的环境配置”介绍全局环境配置和用户个人环境配置。

第 9 章“Linux 文件系统管理”介绍 Linux 默认的文件系统 ext2 与 ext3 的结构与基本管理。

第 10 章“磁盘分区的创建与挂载”介绍磁盘分区的创建、格式化、文件系统检测与挂载操作。

第 11 章“分区文件系统的管理”介绍磁盘文件系统、磁盘的分区格式化操作以及 quota 磁盘配额管理。

第 12 章“文件系统的归档管理”介绍文件的打包、压缩与备份等文件的过程化处理操作。

第 13 章“软件系统扩充”介绍 Linux 操作系统中的软件控制方法，包括 Tallbar 与 RPM 软件管理方法。

第 14 章“Linux 的进程管理”介绍 Linux 操作系统中的进程管理方法和计划任务配置方法。

第 15 章“Linux 的启动引导器”介绍 Linux 操作系统的启动机制，启动引导器 GRUB 的配置方法。

第 16 章“Linux 的启动与服务”介绍服务的启动机制与启动控制。

第 17 章“Linux 的基本网络配置”介绍 Linux 网络的基本概念、网络的工作机制与配置方法。

第 3 篇“Shell 基础”包括以下两章:

第 18 章“Shell Script 基础”介绍 Shell 脚本的结构与基本要素。

第 19 章“Shell Script 中的结构控制语句”介绍 Shell 脚本中常见的条件判断和循环控制语句。

第 4 篇“网络服务基础”包括以下 5 章:

第 20 章“NFS 网络文件系统”介绍 NFS 服务的工作原理以及 NFS 服务器的搭建、配置与管理。

第 21 章“Samba 服务的配置与应用”介绍 Samba 服务的工作原理以及 Samba 服务器的搭建、配置与管理。

第 22 章“DNS 服务器的基本配置”介绍 DNS 服务的工作原理以及 DNS 服务器的搭建、配置与管理。

第 23 章“Web 服务的配置与应用”介绍 Web 服务的工作原理以及 Apache 服务器的搭建、配置与管理。

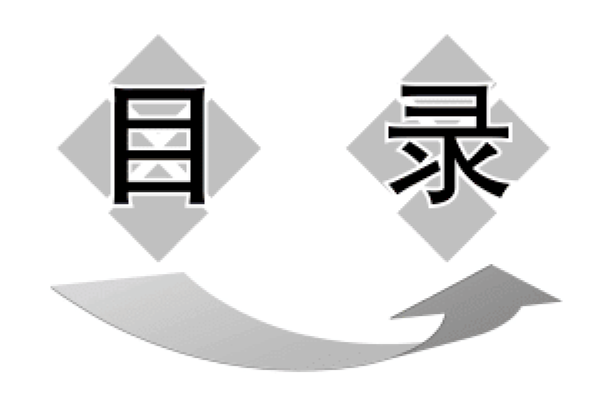
第 24 章“远程管理工具的管理与使用”介绍 SSH 服务器的搭建、配置与管理以及非对称加密认证。

全书由李芳晴女士整体负责策划，北京盛浩博远教育科技有限公司的教研团队李善军先生和郭文明先生编著。在此特别感谢清华大学蔡莲红教授在成书过程中所给予的指导和帮助。

Linux 操作系统体系涉及的范围广泛、结构复杂，同时这个操作系统也在不断更新，以便容纳更多的新技术与新功能，因此本书难免会有错误和遗漏，敬请读者批评指正。

编者

2012 年 3 月



第 1 篇　Linux 操作系统基础

第 1 篇　Linux 操作系统基础

对于操作系统知识的学习而言，最好的方法就是在操作系统环境中不断地调试与摸索。本篇将引导读者建立 Linux 操作系统，认识 Linux 操作系统的基本使用环境，熟悉 Linux 操作系统中的基本操作，以便更好、更快地认识与熟悉这个操作系统。

第1章 Linux 概述

作为一本介绍 Linux 操作系统的教材，本书遵循循序渐进的原则。当前 Linux 的发展是迅速的，面对分纷繁复杂的 Linux 版本和一些既熟悉又陌生的相关词汇，应该如何尽快地理解和把握其发展脉络呢？这就是本书第 1 章所要解决的问题。

本章主要介绍 Linux 的背景知识。涉及 Linux 的产生和发展、Linux 发行版以及 Linux 与 GNU 的关系。诚然，Linux 的历史是一个话题丰富的内容，本章只想用最简明的论述为读者揭开 Linux 的第一层面纱，从而使读者对 Linux 的背景有一个大概的认识。

Linux 是一种应用于 PC 和工作站上的操作系统，是一种功能完善、性能稳定、成本低廉的优质操作系统。Linux 是在 20 世纪 90 年代早期由 Linux Torvald 和分布于世界各地的其他程序员共同开发的。作为一种操作系统，Linux 提供的许多功能都与 UNIX、MacOS 和 Windows NT 相同。然而，与其他操作系统相比，Linux 因其在功能和灵活性方面的优势而更加独特。

大多数 PC 操作系统，如 Windows，都是针对功能有限的 PC 开发的。当这些 PC 发展成为日常生产、生活中的基本工具时，与之相适应的操作系统便开始了不断的更新过程，这种更新最根本的目的就是要使操作系统本身能更好地适应 PC 硬件能力的提高。与上述操作系统不同的是，Linux 是在一种完全不同的环境中被开发的。Linux 是应用在服务器和小型机的 UNIX 操作系统的 PC 版本，它将 UNIX 操作系统的速度、效率和灵活性带给 PC，同时还充分利用了 PC 的硬件能力。

Linux 操作系统的网络能力继承了 UNIX 操作系统的高效、稳定的特点。具有安装、支持和维护一个功能完整的网络所需的一切特性。

1.1 操作系统和 Linux

操作系统是管理计算机的软件和硬件资源的一种程序，是计算机中最基本的应用程序。

如果从过程的角度去观察整个计算机系统，可以发现人对计算机的使用实质上是人对计算机中的操作系统的使用（如图 1.1 所示）。人借助于操作系统管理计算机中的软件和硬件资源，借助于操作系统向计算机发出计算指令并得到输出结果。

操作系统的功能传统上包括 3 个方面：文件管理、程序管理以及与用户之间的交互。

文件管理是操作系统对计算机中的硬件资源和文件系统的管理（如图 1.2 所示）。操

作系统对计算机中的硬件资源的管理是通过驱动程序来实现的，所以很多操作系统都将计算机中的硬件设备当做设备文件来对待（设备驱动程序本身就是一种文件），称为设备文件以与文本文件相区别。

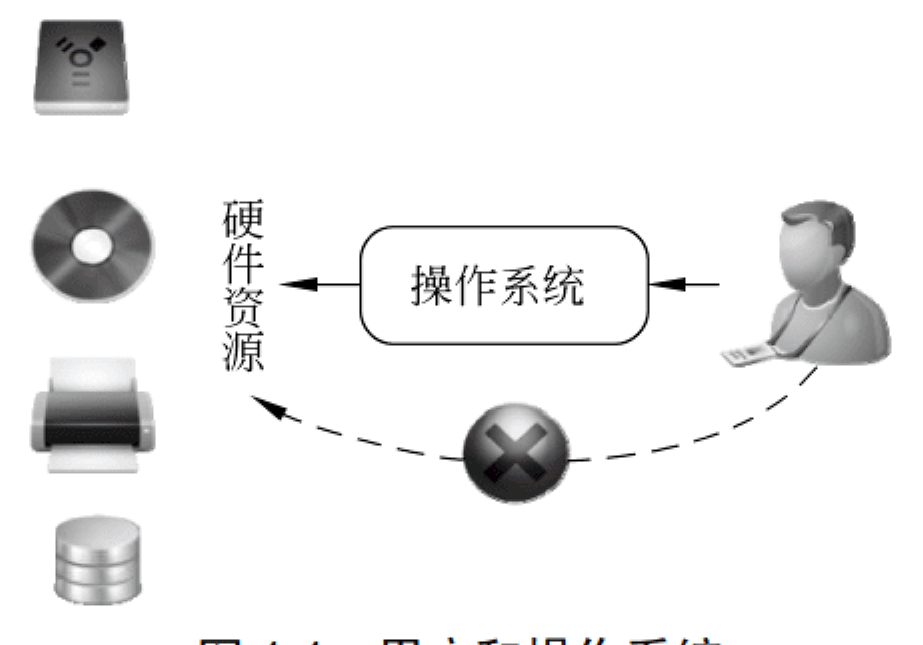


图 1.1　用户和操作系统

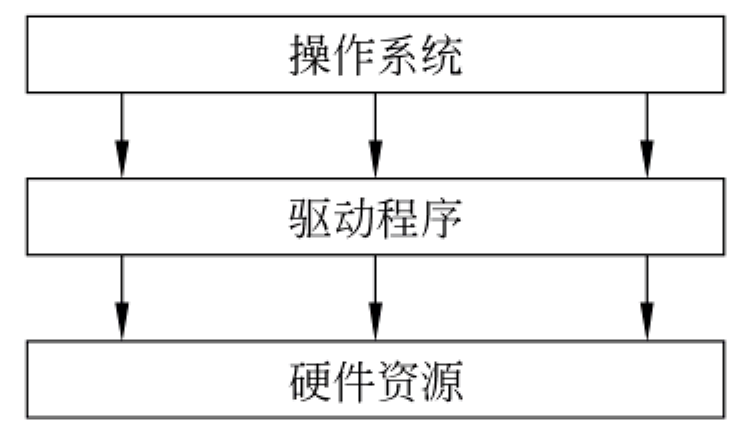


图 1.2　操作系统与硬件资源

程序管理是操作系统的另一个重要功能。一个应用程序如果需要在计算机中被使用，首先必须被加载到计算机的内存中，然后由 CPU 执行程序的功能。操作系统控制了所有程序的加载和执行过程。

与用户之间的交互是操作系统的又一个重要功能（如图 1.3 所示）。用户所要求的各种操作任务实际上最终均是由计算机的硬件设备完成的，而硬件资源是受操作系统内核控制的。操作系统内核仅能识别 0、1 这种二进制值。如何将用户输入的复杂的计算要求解析为操作系统内核可以理解的二进制代码，以及如何将内核的运算结果值解析为用户能够理解的字符代码就是操作系统必须要实现的功能。

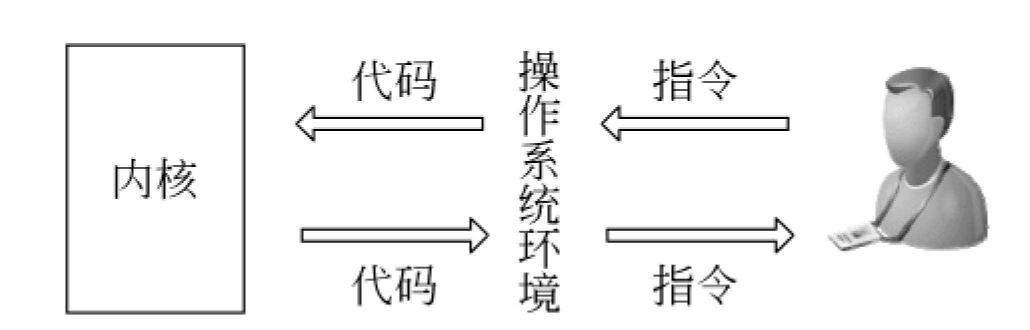


图 1.3　操作系统环境负责命令解释

从以上 3 个层面上看，操作系统可以被理解为是连接用户与计算机硬件之间的桥梁，是用户与硬件之间的界面。

文件管理、程序管理以及与用户之间的交互都是操作系统共同拥有的传统特性。而 Linux 如同所有的 UNIX 版本一样，又增加了两个特性：它是一种多用户和多任务的系统。

（1）作为一种多任务系统，系统可同时执行多个任务，即在一个任务执行的同时可以启动另一个任务开始工作。

（2）作为一种多用户系统，多个用户可以同时登录到系统中，并且每个用户都可以通过自己独立的终端与系统进行交互，使用系统资源。

最初设计操作系统是为了充分利用硬件设备的功能。随着硬件性能的提高，操作系统自身也在不断地完善以便充分发挥硬件设备的计算能力。从这个角度看，操作系统是根据硬件能力而不是用户需求来设计的。因而，操作系统更倾向于是稳定的、不可变的，并要求用户遵循硬件功效的设计。

而另一方面，Linux 是一种灵活可变的操作系统，从而反映出它根源于 UNIX 的特点。作为一种类 UNIX（UNIX-like）操作系统，Linux 具有与 UNIX 类似的灵活性，它

能灵活地调度各种不同性质的任务、灵活地定制操作系统功能。在大型数据计算、图形图像处理、网络数据通信等众多领域，我们都能见到 Linux 的身影。这种灵活性是在充分利用计算机硬件性能的基础上实现的，是性能与效率的综合体现。

1.2 UNIX 简介

Linux 源于 UNIX 操作系统，是 UNIX 操作系统的一个分支，因此有必要了解一些与 UNIX 操作系统有关的背景知识。

UNIX 操作系统的历史漫长而曲折，它的第一个版本是在 1969 年由 Ken Thompson 在 AT&T Bell（贝尔）实验室实现的，运行在一台 DEC PDP-7 计算机上。

这个操作系统非常粗糙，与现代 UNIX 相差很远，它只具有操作系统最基本的一些特性。尽管这样，这个操作系统还是在 Bell 实验室内部的研究人员中被使用，这个使用过程可以被看做是 UNIX 的一个不断完善的过程。

1973 年，Dennis Ritchie 和 Ken Thompson 合作使用 C 语言对整个 UNIX 系统进行了再加工和编写，使得 UNIX 能够很容易地被移植到其他硬件的计算机上。从那以后，UNIX 就开始了令人瞩目的发展。图 1.4 描绘了 UNIX 操作系统演变的一个基本过程。

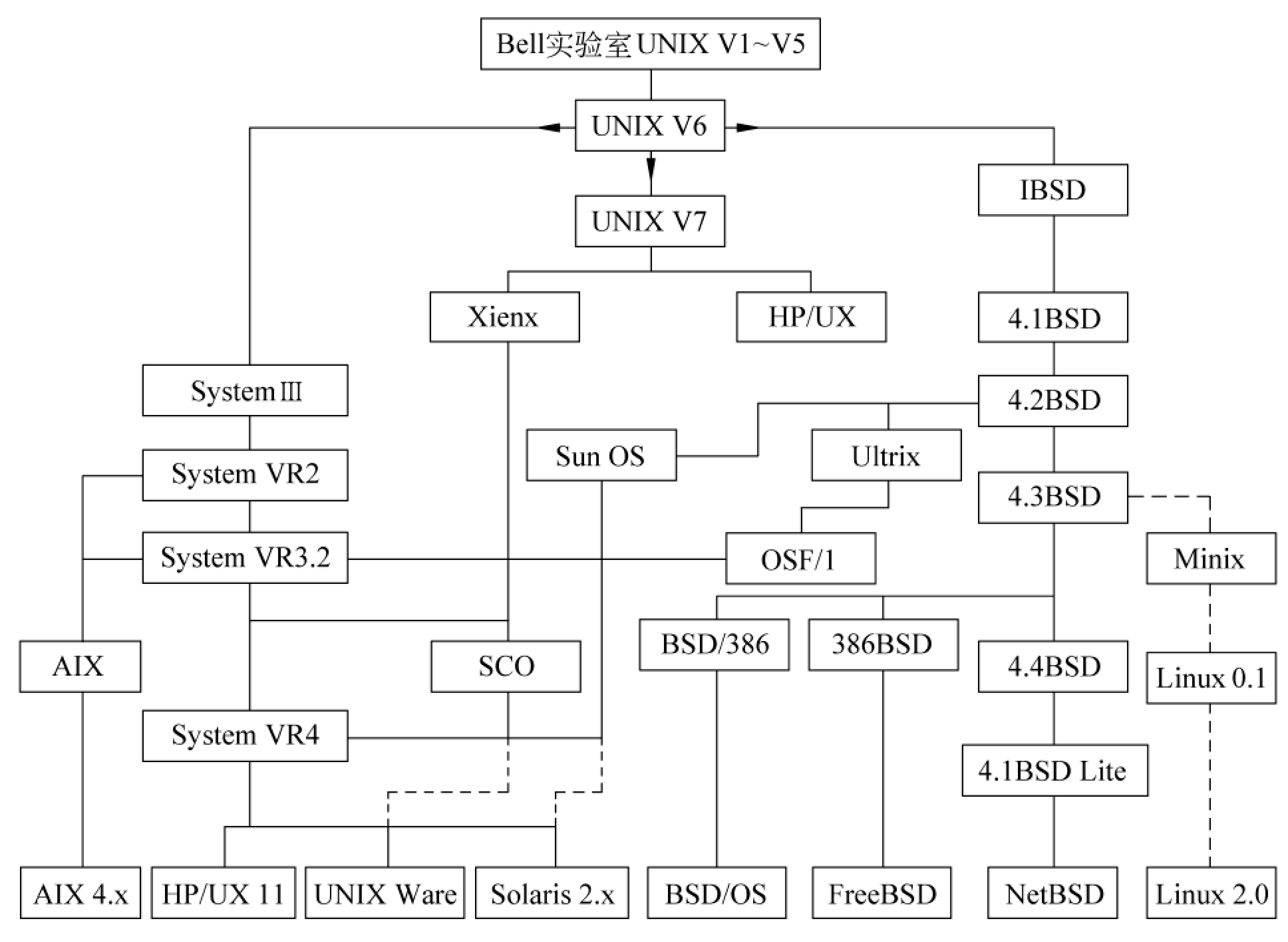


图 1.4　UNIX 发展路线图

1973 年时 AT&T 还没有把 UNIX 作为他的正式商品，研究人员只是在实验室内部使用并完善它。作为一个研究项目，UNIX 在一些科研机构和大学也受到了广泛的追捧，利用 UNIX 来论证操作系统机制，研究操作系统发展方向成为当时很热门的一种研究方法。而 AT&T 也顺应发展的需求，以分发许可证的方法对 UNIX 收取很少的费用，就能

使大学和研究机构获得 UNIX 的源代码以进行研究。这样，UNIX 的源代码就被散发到了各个大学中了。这样一方面使得 UNIX 被移植到了各种不同的硬件环境中，另一方面也培养了广泛的 UNIX 用户群。

由于操作系统的开发相当困难，而此时的 UNIX 不需要太多的花费，因此很多计算机厂商选择了 UNIX 作为他们生产的计算机使用的操作系统。他们把 UNIX 移植到自己的硬件环境下，而不必从头开发一个操作系统。

到了 20 世纪 70 年代末，在 UNIX 发展到了版本 6 后，AT&T 认为 UNIX 已经成熟，并形成了可持续发展的市场价值，因此成立了 UNIX 系统实验室（UNIX System Lab，USL）来继续发展 UNIX。至此开始了 UNIX 全面商业化的进程，UNIX 从一个可以很轻易得到的产品转变为一个付费的操作系统，而且价格不菲。在计算机网络发展得如火如荼的今天，UNIX 仍然是一个高端、高成本的操作系统。

当前，UNIX 操作系统主要应用于高端服务器领域。在生产环境中，高端服务器通常由小型计算机和中型计算机来充当。常见的高端服务品牌如下。

（1）IBM 公司的 P 系列产品，流行的包括 P5 570、P6 550、Power 750 等产品。

（2）SUN 公司的 SPARC 系列产品，流行的包括 T5 120 和 M4000 等产品。

（3）HP 公司的 Proliant 系列产品，流行的包括 DL 和 ML 系列产品。

（4）曙光公司的曙光天阔系列产品。

有关于服务器硬件产品的信息，读者可自行参考相关资料或厂商站点，本书不做详细讨论。

对于不同的服务器硬件厂商而言，一般都有自有品牌的 UNIX 操作系统供自己的产品使用。这些 UNIX 操作系统的起源均可追溯到 UNIX V6，是在 UNIX V6 的基础上衍生出来的产品（如图 1.4 所示）。所以，从当前 UNIX 操作系统的发展来看，UNIX 已不再是指某一个操作系统了，而是一系列操作系统的总称。

当前，活跃在生产一线的 UNIX 产品主要包括：

（1）Free BSD。是源于美国加州大学伯克利分校开发的 BSD 系统的一个重要的 UNIX 分支。FreeBSD 所支持的平台依据支持程度分成 4 个等级。第一线平台（Tier 1，完整支持平台）目前包括 i386、Sparc 64、AMD 64 及 PC 98。第二线平台（Tier 2，发展平台）包括 PowerPC 及 IA64。对于第一线与第二线平台，FreeBSD 会支持维护与稳定性，同时大多数的新功能也都会被要求在这些平台上能够正常运作。第三线平台（Tier 3，实验平台）目前只包括了 S/390，这个等级的平台不被 FreeBSD 正式支持。而其他的平台都被归类到第四线平台。

（2）AIX。是 IBM 开发的一套 UNIX 操作系统。它符合 Open Group 的 UNIX 98 行业标准（The Open Group UNIX 98 Base Brand），通过全面集成对 32 位和 64 位应用的并行运行支持，为这些应用提供了全面的可扩展性。它可以在所有的 IBM P 系列和 IBM RS/6000 工作站、服务器和大型并行超级计算机上运行。

（3）Solaris。是 Sun Microsystems 开发的 UNIX 操作系统。具有优良的系统架构，目前 Solaris 属于混合开源软件，支持 Intel X86 平台。

（4）HP-UX。是 HP 公司开发的 UNIX 操作系统，可以在 HP 的 PA-RISC 平台、Intel

的 Itanium 平台上运行。

在基于 UNIX 的系统中最激动人心和最具有挑战性的发展也许就是 Linux。作为 UNIX 的一种衍生版本，Linux 正以其稳定及包容的特性出现在各种各样的应用环境中。

1.3 Linux 的产生和发展

1.3.1 Linux 产生的时代背景

Linux 出现在 1991 年，其作者是芬兰赫尔辛基大学的学生 Linus Torvalds，Linux 在 1991 年这个时间出现是有一定的原因的。在 20 世纪 80 年代，计算机硬件的性能在不断地提高，PC 的市场在不断地扩大，而当时可供计算机选用的操作系统主要有 UNIX、DOS 和 MacOS 这几种。其中 UNIX 价格昂贵，不能运行于 PC。而 DOS 显得十分简陋，且源代码被软件厂商严格的保密。MacOS 是一种专用于苹果计算机的操作系统。这时，计算机科学领域迫切需要一个更为完善、强大、廉价和完全开放的操作系统。

在这种需求背景之下，计算机科学领域出现了一系列创新性的活动，致力于开创一个能满足不同需求的、高效的、完全开放的应用体系。其中由 Richard Stallman 倡导和开创的 GNU 项目很好地反映了时代的需求，他出色地组织和发展了一系列的开放源代码软件，为最终实现开放源代码的操作系统奠定了基础。

Richard Stallman 是 MIT（人工智能实验室）的一名工程师，其开创的 GNU 项目是一个致力于推出自由、高质量软件的运动。GNU 一词是 GNU's Not UNIX 的词首字母缩略语，他针对 UNIX 对源代码保密和收取高额费用，倡导软件和其他产品不同，在复制和修改方面它不该受到任何限制。只有这样，才能开发出更好、更强的软件。1983 年，他在著名的《GNU 宣言》中向世人宣告了 GNU 项目的启动，开始了贯彻其哲学思想的自由软件运动。

GNU 项目在提倡开放源代码的同时也强调软件版权的保护。GNU 定义了 GPL（General Public License，通用公共许可证），该许可协议在保证使用者共享和修改开源软件的自由的同时，定义了使用者对开源软件的义务。一个软件程序一旦加入了 GPL 协议，该软件将受到 GPL 协议的约束和保护。GPL 协议从出现至今经历了多次修改，目前最新的 GPL 协议的版本是 GPL 3.0。基于 GPL 协议的发布的软件有以下特点。

（1）任何人都拥有运行该软件的权利和自由，而且可以用于任何目的。

（2）任何人都拥有修改该软件以适应个人需要的权利。

（3）任何人都拥有再发行拷贝的权利，可以是无偿的，也可以收费。

（4）任何人都拥有发行该软件修改后的版本的权利，从而使社区可以从所作的改进中获益。

任何基于 GPL 协议发布的软件在经过修改后也必须基于 GPL 协议发布。例如，Linksys 是思科开发的路由器。该路由器是基于 Linux 内核修改后的版本，根据 GPL 的协议，思科必须公布修改后的版本的源代码。

GNU 项目得到了大量的社区支持，促进了开源软件的发展，但是 Richard Stallman

并没有开发出操作系统，当他正在努力实现他的自由操作系统的时候，Linux 已经如火如荼地发展起来了。Linux 符合自由软件的精神，成为 GNU 社区中重要的操作系统。

1.3.2 Linux 的产生和发展

20 世纪 80 年代末，由于供教学使用的典型操作系统很少，当时在荷兰当教授的美国人 Andrew S. Tanenbaum 从零开始编写了一个操作系统，名为 MINIX。MINIX 以 Intel 8086 微处理器为基础，其初衷是为了向学生讲授操作系统的内部工作原理。

作为一个操作系统，MINIX 算不上一流，但它的好处是公开源代码。在 Tanenbaum 写的《操作系统：设计与实现》这本书中给出了 12 000 行用 C 语言和汇编语言写的源代码。程序员或黑客第一次可以有机会读一读操作系统的源码——这种被软件商严加看管的东西。Tanenbaum 用简洁的笔触详尽探讨了编写操作系统的艺术。“他是个一流的作者，迷住了一批当时计算机领域最聪明的大脑。” 全世界学计算机的学生都在钻研这本书，通过读它的源代码来了解他们电脑里运行的 MINIX 操作系统。 Linus Torvalds 就是这些学生中的一个。

在吸收了 MINIX 精神的基础上，Linus 于 1991 年 9 月中旬写出了属于自己的 Linux 操作系统，版本为 Linux 0.01。这个版本是在综合了 MINIX 新闻组上对 MINIX 的优缺点的建议而实现的，虽然这一版本的 Linux 很小，但它是 Linux 时代开始的标志。

Linux 得到了来自世界各地的程序员的帮助，Linux 的源代码通过在芬兰和其他一些地方的 FTP 站点传遍了全世界。在很短的时间内版本被不断地更新，一个严谨的、完善的操作系统在遍布于世界各地的程序员的协作下通过互联网逐渐地呈现在人们的面前。

Linux 与 MINIX 的关系可谓紧密，可以说 Linux 是在 MINIX 思想的引导下完成的，尽管 Linux 并没有照搬 MINIX 代码；而 MINIX 虽然是 Tanenbaum 独立完成的作品，但 Tanenbaum 借鉴了 UNIX 操作系统的设计思想，确切地说是借鉴了 4.3BSD 这个 UNIX 操作系统分支的设计思想。这样一来，使得 Linux 与 UNIX 存在了某种共性，以至于发展到后来 Linux 被当做了 UNIX 的一个分类，归为类 UNIX 操作系统的行列。

需要注意的一点是，这里说的 Linux 仅表示的是操作系统的内核。如同 UNIX 一样，Linux 通常被分割成 3 个主要部分：内核、环境和文件系统。内核是运行程序和管理硬件设备的核心程序，是整个操作系统的灵魂。环境则为用户提供了一个界面，他接受来自用户的命令，同时将这些命令发送到内核去执行。文件系统对存储在一个存储设备中的文件进行组织。内核、环境和文件系统这三者合在一起才能形成一个基本的操作系统，而这里说的 Linux 仅仅是指操作系统中的内核。

到 2011 年 8 月，Linux 内核的最新版本为 2.6.36。Linux 内核的最新版本可以访问 http://www.kernel.org 站点获得。

1.3.3 Linux 发行版

Linux 发行版（Distribution）是指将 Linux 内核、环境和文件系统按一定的需求组合起来，添加一定的应用功能而形成的一个完整的操作系统。

Linux 发行版常常由软件供应商制作并供分享和销售，目前 Linux 的发行版已超过 100 种，各自实现的功能和目标定位不尽相同。下面向读者介绍几种常见的 Linux 发行版。

1. Debian

Debian 最早是由 Lan Murdock 于 1993 年创建的，其大部分的基本工具均来自 GNU 计划，可以算是迄今为止最严格地遵循 GNU 规范的 Linux 系统。它以严谨、开放和自由而著称，秉承 Linux 网络协作开发的精神，是目前著名的 Linux 发行套件中唯一的非商业性版本。

Debian 系统包括 3 个分支版本（branch）：稳定版（stable）、测试版（testing）和不稳定版（unstable）。

unstable 为最新的测试版本，其中包括最新的软件包，但是也有相对较多的错误（bug），适合桌面用户。unstable 版的版本代号永远都被称为 sid。

testing 版都经过了对 unstable 的测试，相对较为稳定，也支持了不少新技术（如 SMP 等）。截至 2011 年 8 月测试版版本代号是 squeeze。

stable 版一般用于服务器，上面的软件包大部分都比较过时，但是稳定性和安全性都非常高。截至 2011 年 8 月 stable 版本号是 5.0.6，开发代号为 lenny，已经于 2010 年 9 月 4 日发布。

有关 Debian 发行版的详细信息可参考 http://www.debian.org/ 站点的信息。

2. Ubuntu

Ubuntu 是基于 Debian 的，旨在创建一个可以为桌面和服务器提供一个最新且一贯的 Linux 系统。Ubuntu 囊括了大量精挑细选自 Debian 发行版的软件包，同时保留了 Debian 强大的软件包管理系统，以便简便地安装或彻底地删除程序。与大多数发行版附带数量巨大的可用可不用的软件不同，Ubuntu 的软件包清单只包含那些高质量的重要应用程序。

Ubuntu 提供了一个健壮、功能丰富的计算环境，既适合家用又适用于商业环境。该项目花费了大量的时间，努力精益求精，每 6 个月就会发布一个版本，以提供最新、最强大的软件。Ubuntu 支持各种形形色色的架构，包括 i386（386/486/Pentium(Ⅱ/Ⅲ/4)和 Athlon/Duron/Sempron 处理器）、AMD 64（Athlon 64、Opteron，最新的 64 位 Intel 处理器），以及 PowerPC（iBook/Powerbook、G4 和 G5）等。

Ubuntu 包括桌面版、上网本版和服务器版 3 个分支版本。

3. Slackware

Slackware 是由 Patrick Volkerding 于 1992 年创建的，是历史最悠久的 Linux 发行版之一。Slackware 发行版坚持追求操作系统的效率最大化，所有的配置均需要通过配置文件来完成。

Slackware 稳定、安全，严格遵守 UNIX 规范，有大批的用户群。Slackware 尽量采

用原版的软件包而不进行任何修改，所以产生新错误的几率很低。Slackware 的版本更新周期较长（大约 1 年），但是新版本的软件仍然不间断地提供给用户下载。

4. SUSE

SUSE 源于德国的著名的 Linux 发行版，在全世界享有较高的声誉。SUSE 自主开发了软件包管理系统 YaST，是一个非常专业、优秀的发行版。

其发行版中个别版本存在收费的情况。SUSE Linux 10.2 版本以后改名为 openSUSE。

SUSE 于 2003 年年末被 Novell 收购。

5. Fedora

Fedora 项目是由 Red Hat 赞助，由开源社区与 Red Hat 工程师合作开发的项目统称。

Fedora Core X 可以看做是 Red Hat Linux 的免费桌面版本，而 Red Hat Linux 的企业版将提供收费服务。

截止 2011 年 8 月，Fedora 的最新发行版为 Fedora Core 14。

6. CentOS

CentOS 是 Red Hat 企业版 Linux 源代码再编译的产物，而且在 Red Hat 企业版 Linux 的基础上修正了不少已知的错误，相对于其他 Linux 发行版，其稳定性值得信赖。CentOS 是本书采用的示例操作系统，由于它是 Red Hat 企业版 Linux 的再编译版本，所以相关的版本信息可参考 Red Hat 发行版的信息。

7. Red Hat

Red Hat 是全球最大的开源技术厂家，其产品 Red Hat Linux 也是全世界应用最广泛的 Linux，也是本书所采用的 Linux 发行版。

对于 Red Hat 而言，其产品包括多个方面，如企业服务器产品、虚拟化产品、中间件产品等。

Red Hat Enterprise Linux 产品包含了以下 3 个子版本：

（1）Red Hat Enterprise Linux AS

（2）Red Hat Enterprise Linux ES

（3）Red Hat Enterprise Linux WS

Red Hat Enterprise Linux AS（AS：Application Server，应用服务器版）是 Red Hat Enterprise Linux 家族中功能最完善的版本，是构建主要操作系统和企业级解决方案的首选产品。与其他版本相比，Red Hat Enterprise Linux AS 包括了最全面的支持服务，能够支持多处理器和大内存的最大型服务器架构，是使 Red Hat Enterprise Linux 成为大型企业部门级服务器和计算中心的最佳解决方案。该产品当前的最新版本为 Red Hat Enterprise Linux AS 5。

Red Hat Enterprise Linux ES（ES：Enterprise Linux，企业服务器版）为初级至中级的服务器系统提供核心操作系统和网络基础设施。Red Hat Enterprise Linux ES 最适合用

在建设网络、文件服务器、打印服务器、邮件服务器、网站服务器和基本商业应用中。Red Hat Enterprise Linux ES 能充分地与 Red Hat Enterprise Linux 家族的其他成员兼容并提供稳定、高效能的服务和良好的技术支持。

Red Hat Enterprise Linux WS（WS：WorkStation，工作站版）是充分地与 Red Hat Enterprise Linux 家族的其他成员兼容和相互补充的产品。Red Hat Enterprise Linux WS 支持最大两个 CPU 的桌面级应用。Red Hat Enterprise Linux AS 的一些应用程序不包括在 Red Hat Enterprise Linux WS 内。

以上介绍了关于 UNIX 的历史和发展过程以及 Linux 的一些背景知识，便于大家对 Linux 的背景有一个大概的了解。需要提醒读者的是，Linux 与 UNIX 的关系是非常紧密的，更多关于 UNIX 操作系统的发展过程和版本结构的知识可自行参阅其他资料。

第 2 章 Linux 操作系统的部署

本章介绍 Linux 操作系统的部署。对于一个新接触的操作系统而言，快速掌握它的最好的方法就是使用它，而我们现在所要做的就是把这个操作系统安装到计算机上。

当前 Linux 操作系统的安装过程越来越人性化，这得益于功能越来越强大的安装程序。这些安装程序可以自动检测硬件设备并尽量去匹配驱动程序。本书采用的是 CentOS 5.5 这个 Linux 发行版，该发行版来自 Red Hat Enterprise Linux 依照开放源代码规定公开的源代码编译而成，表现出了与 Red Hat 相同的管理结构和实现机制，充分体现了流行的 Linux 操作系统的管理和使用风格，掌握该发行版的使用方法对于使用其他发行版是大有裨益的。

这里将从硬件设备的规划谈起，循序渐进地引导读者完成 CentOS Linux 操作系统的安装。

对于 Linux 操作系统而言，读者可以在一台裸机上完成对它的安装，也可以在一台已经有操作系统（如 Windows 操作系统）的计算机上进行安装，从而实现多系统环境，当然，这是在我们的学习环境中的部署策略。在实际的生产环境中，操作系统的安装是一个很重要的工作，往往会使用专门的服务器计算机来安装这个操作系统。

2.1 安装前的准备

在安装一个操作系统之前，首先要了解当前的硬件环境是否能够满足操作系统的安装需求，还需要明确安装前应该为安装作哪些准备。

2.1.1 Linux 系统的硬件需求

对于 Linux 操作系统而言，当其实现的功能不同时其对硬件的要求也不尽相同。当仅需要将 Linux 计算机当做终端使用，并且不安装 X Window 系统（图形环境）时，这时 Linux 需要的硬件级别很低。而当需要将 Linux 计算机当做服务器来使用时，或者需要使用 X Window（图形环境）时，那么，就需要使用高性能的硬件配置了，尤其是内存（RAM）的容量、显存与硬盘的容量等关键设备。

事实上，Linux 操作系统并未对主机的硬件配置表现出明确的要求。不同发行版提出的硬件需求也仅仅是为使用者提供的一个硬件规划建议，笔者也曾在一台 486 等级的主机上安装了 Linux 操作系统作为工作环境中的邮件服务器使用，至今仍然运行良好。

从当前的计算机硬件环境来看，绝大多数主机都能满足 Linux 的硬件需求。

需要注意的一点是，由于 Linux 内核对于硬件驱动程序的更新速度可能会落后于硬件产品的开发速度。所以，当硬件设备中存在一些最新的硬件设备时，可能会发生操作系统不能自动识别的情况。这时，需要使用者自行获取该硬件设备的驱动程序并进行安装。

通常 Linux 发行商在发布 Linux 前会针对该版本 Linux 所预定支持的硬件设备进行说明，该说明除可以在 Linux 的 Howto 中查询外，也可以到各个相关的 Linux 发行版的官方站点查询。表 2.1 列出了常用的 Linux 硬件支持站点。

表 2.1　常用的 Linux 硬件支持站点

站 点 名 称	网　　址
Linux 的硬件 How to	http://www.linux.org/hardware/
Red Hat 的硬件支持	http://hardware.redhat.com/hcl/?pagename=hcl
SuSE 的硬件支持	http://hardware.suse.de/index.php?LANG=en_UK
Linux 对打印设备的支持	http://www.linuxprinting.org
Linux 对笔记本计算机的支持	http://www.linux-laptop.net/
显卡对 XFree86/Xorg 的支持	http://www.linuxhardware.org

虽然 Linux 操作系统对硬件资源的要求不是十分严格，但为了能充分发挥系统的性能，在配置硬件资源时也应该为 Linux 留出足够大的扩充余地。总之，大家要记住一个原则，Linux 可以包容低级别的硬件，但硬件等级越高越能发挥 Linux 操作系统的性能。

2.1.2　明确当前系统的硬件信息

尽管 Linux 的安装程序会自动识别并驱动硬件设备，但还是会有例外，特别是当你所采用的硬件设备比较陈旧或比较新时，都会导致安装程序无法识别的问题出现。所以在安装系统之前应该对当前计算机的硬件设备有一个大概的了解。具体包括以下设备信息。

（1）CPU 信息：当前主机所使用的 CPU 的架构与型号。Linux 操作系统支持多种架构的 CPU（如 Alpha、SPARC、PowerPC 和 Xeon），但常用的是 Intel 公司和 AMD 公司的基于 X86 架构的 CPU。

（2）内存信息：主要是内存容量信息。

（3）硬盘信息：包括硬盘的容量信息、硬盘驱动器的接口信息（IDE 接口、SATA 接口还是 SCSI 接口）、硬盘的现有分区信息以及硬盘数量信息。

（4）CD-ROM/DVD-ROM 信息：主要是设备的接口信息（IDE 接口、SATA 接口还是 SCSI 接口）。

（5）鼠标和键盘的信息：包括鼠标和键盘的接口信息以及键盘的布局信息。

（6）网络接口卡的信息：包括网卡的型号和速率，并规划好网卡的 IP 地址和网关等相关配置。若无法确定，可暂时使用网络中的 DHCP 服务器自动分配的地址信息。

（7）显卡的信息：包括显卡的型号和显存的容量信息。

对于这些硬件信息，可以通过硬件产品附带的手册来获得，也可以通过本机的 BIOS 来获取相关信息。另外，还可以通过 Windows 操作系统获得硬件信息。

2.1.3　Linux 中的存储设备编号

Linux 的安装实际上是将 Linux 操作系统安装到硬盘中的过程，在 Linux 操作系统中对硬盘的表示方法和对硬盘分区的表示方法是有其特殊的规定的，下面首先介绍设备的表示法，以明确安装位置这一重要信息。

在 Linux 操作系统中，所有的硬件设备都是以文件的形式存在的，即实际的硬件设备在系统中表现为一个文件，管理员对设备的指定与控制也是通过文件实现的（如图 2.1 所示）。这一点很重要，使得管理员通过对文件这种直观的对象操作就可以实际控制设备了，设备文件实际上为管理员提供了一种控制实际物理设备的方法。在 Linux 系统中，设备文件均存放在/dev 目录下。下面重点介绍几个常用的硬件设备文件。

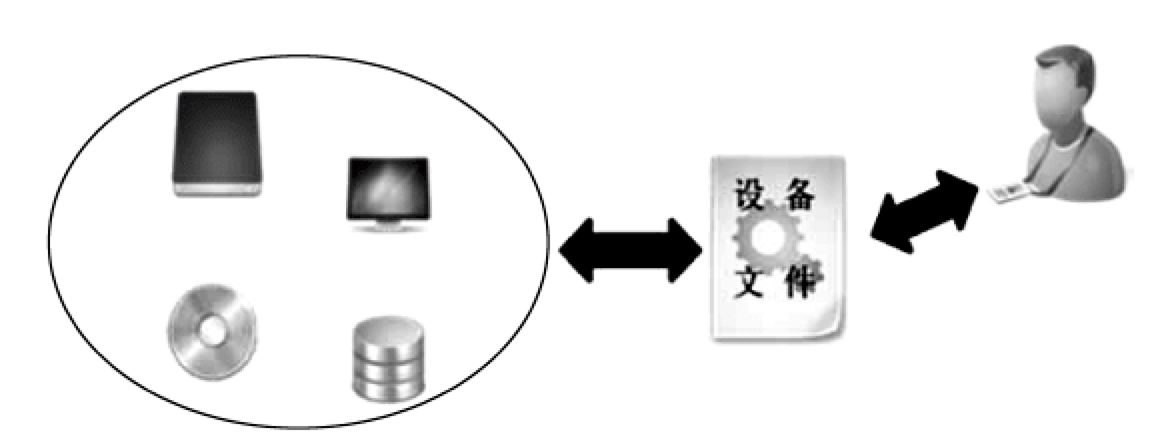


图 2.1　硬件设备以文件的形式供系统调用

1. IDE 接口设备的表示方法

在 PC 中，硬盘的接口通常有 3 种：IDE 接口、SCSI 接口和 SATA 接口。下面首先讨论 IDE 接口硬盘的表示方法。

主板上的 IDE 接口在标准的情况下有两个，IDE1 和 IDE2。每个 IDE 接口上可再接两个 IDE 接口的硬盘或光驱。主板上的 IDE 接口结构如图 2.2 所示。

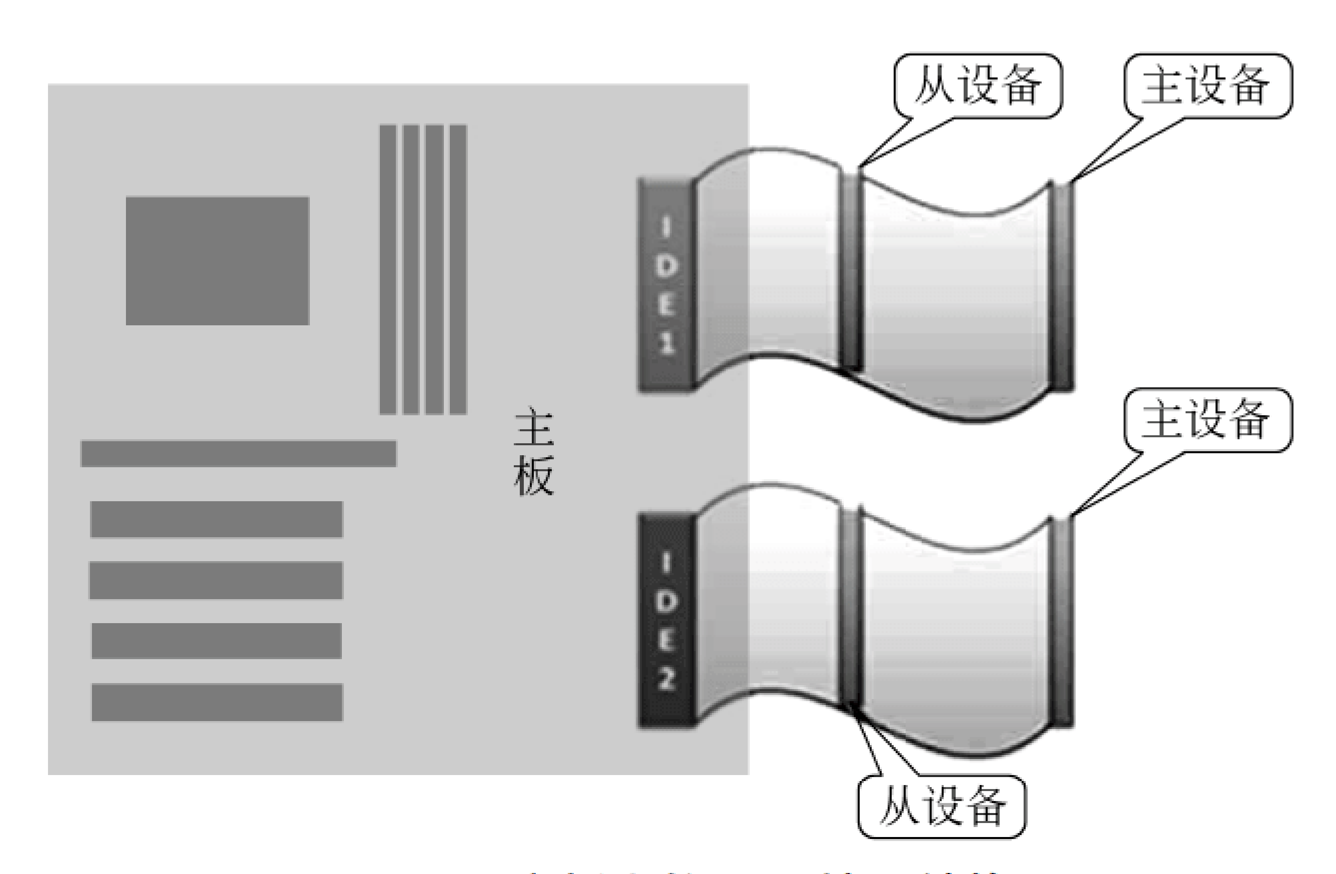


图 2.2　主板上的 IDE 接口结构

每个 IDE 接口上的两个设备均可分为主设备和从设备，以实现为每个设备分配独立的硬件资源地址（包括 IRQ、I/O 地址以及 DMA 等）。这样，这 4 个 IDE 接口设备就分

别为“IDE1 接口上的主设备”、“IDE1 接口上的从设备”、“IDE2 接口上的主设备”和“IDE2 接口上的从设备”。

Linux 操作系统中对于 IDE 接口设备采用/dev/hd$x$ 这种方法来表示（/dev 表示/dev 目录。hd 是 hard disk 的缩写，表示 IDE 接口硬盘；x 是硬盘的序号，表示第几块 IDE 硬盘），具体到每个设备的表示方法如表 2.2 所示。

表 2.2　IDE 接口设备对应的设备文件

设　　备	Linux 设备文件	设　　备	Linux 设备文件
IDE1 接口上的主设备	/dev/hda	IDE2 接口上的主设备	/dev/hdc
IDE1 接口上的从设备	/dev/hdb	IDE2 接口上的从设备	/dev/hdd

需要注意的是，IDE 接口的设备包括硬盘和光驱两种。表 2.2 中的设备文件不仅指硬盘，如果光驱也为 IDE 接口，则也采用表 2.2 中的设备文件名来表示。

在实际的工作环境中，除 IDE 接口的设备之外，SCSI 接口的设备也十分常见，特别是在服务器设备中采用得非常多。在了解了 IDE 设备的表示方法后，下面介绍 SCSI 设备的表示方法。

2. SCSI 接口设备的表示方法

在计算机中，SCSI 接口的设备是通过 SCSI 接口卡连接的。一块 SCSI 接口卡上可连接多个 SCSI 接口的硬盘或光驱设备。其结构如图 2.3 所示。

在 Linux 操作系统中对于 SCSI 接口设备采用/dev/sd$x$ 这种方法表示（/dev 表示/dev 目录。sd 是 SCSI hard disk 的缩写，表示 SCSI 接口硬盘；x 是硬盘的序号，表示第几块 SCSI 硬盘），具体到每个设备的表示方法如表 2.3 所示。

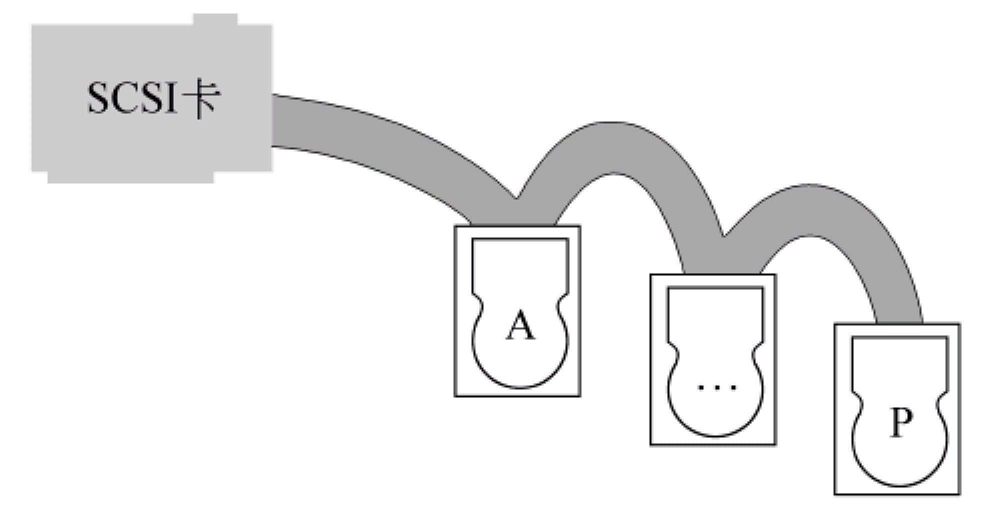


图 2.3　SCSI 接口设备的链接方式

表 2.3　SCSI 接口硬盘对应的设备文件

设　　备	Linux 设备文件
第一块 SCSI 硬盘	/dev/sda
第二块 SCSI 硬盘	/dev/sdb
⋮	⋮

需要注意的是，SCSI 接口的设备包括硬盘和光驱两种。表 2.3 中的设备文件不仅指硬盘，如果光驱也为 SCSI 接口，则也采用表中的设备文件名来表示。

在 Linux 操作系统中，对 SATA 接口硬盘的表示方法与 SCSI 接口的表示方法相同。但如果在 BIOS 将 SATA 接口设置为 IDE 模式运行的话，则 SATA 接口硬盘的文件表示方法就要与 IDE 接口硬盘的文件表示方法相同。

3. 软驱设备的表示方法

在当前的计算机配置中，软盘驱动器已经很少被使用了。这主要因为软盘的容量小、

数据易损坏等缺点。便携式存储设备已经逐渐被 Flash 闪存所代替。出于知识完整性的考虑，这里介绍软驱设备的表示方法。

软盘驱动器与主板的接口为 fd，现在的主板上最多有一个 fd 接口，这个接口上可连接两个软盘驱动器（如图 2.4 所示）。

对于这两个软驱而言，接在 C 位置的软驱使用/dev/fd0 文件来表示，接在 B 位置的软驱使用/dev/fd1 文件来表示。注意，这里的连接位置很重要，仔细观察图 2.4 中的数据线可见，在 C 接口后的数据线被做了个扭接，这个扭接的作用是标识第一个软盘驱动器，这个扭接被称为 α 结，连接在 α 结位置的软驱使用/dev/fd0 文件来表示。

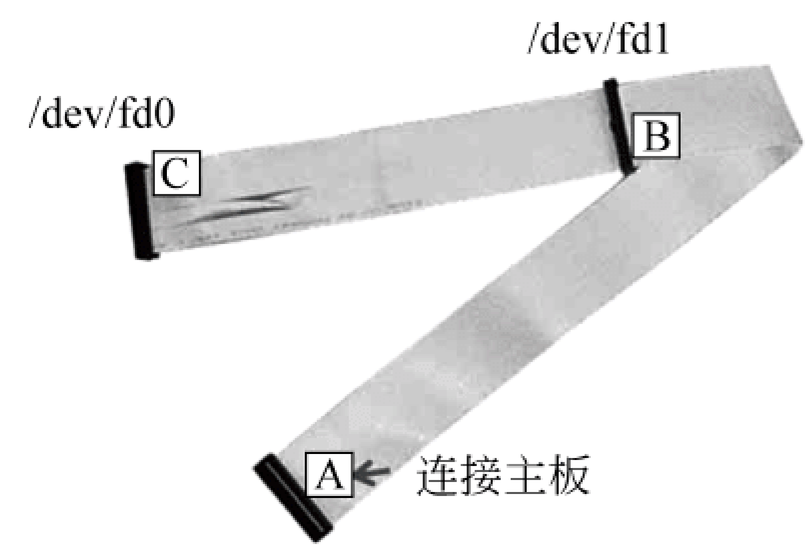


图 2.4　软盘驱动器的表示方法

4. USB 接口设备的表示方法

USB 接口设备在 Linux 中被当做 SCSI 接口设备来表示，即也采用/dev/sd*x* 这种文件表示方式。

2.1.4　Linux 中硬盘分区的表示方法

在介绍完硬盘设备的表示方法之后，需要继续明确硬盘中分区的表示方法。分区是硬盘必要的一种逻辑结构，定义了数据存储的范围。硬盘不能直接用来存储数据，必须对硬盘进行分区后，将数据存储在分区中。

不论是 IDE 接口的硬盘还是 SCSI 接口的硬盘，其分区方式均可以概括为以下 4 种（如图 2.5 所示）：

（1）4 个主分区

（2）3 个主分区+1 个扩展分区

（3）2 个主分区+1 个扩展分区

（4）1 个主分区+1 个扩展分区

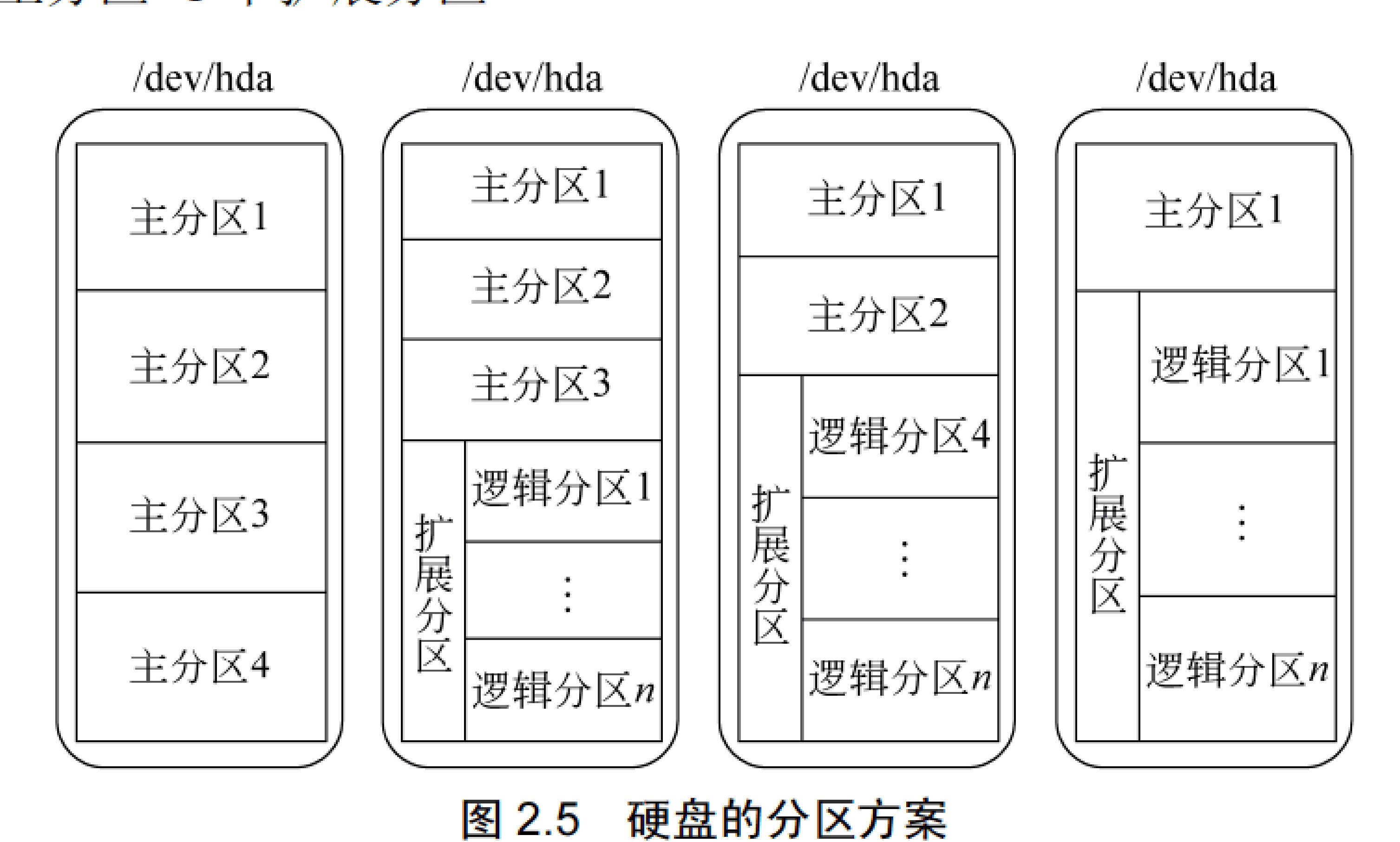


图 2.5　硬盘的分区方案

为什么硬盘会有此种分区方案呢？接下来讨论这个问题。

对于一块硬盘而言，其 0 磁道 0 柱面 1 扇区为引导扇区，称为 MBR（Master Boot Recorder，主引导记录）。MBR 对系统而言是非常重要的，因为它包含了两个重要的信息：

（1）操作系统引导程序。

（2）磁盘分区表（DPT，Disk Partition Table）。

有关于操作系统引导程序的问题将在相关章节中讨论，在这里仅讨论磁盘分区表的问题。

引导扇区的容量为 512B，这里磁盘分区表占用 64B，用于记录磁盘中每个分区的位置。但是，每个分区位置的记录需要占用 16B，这就导致了在磁盘分区表中最大只能记录 4 个分区位置的问题，如图 2.6 所示。

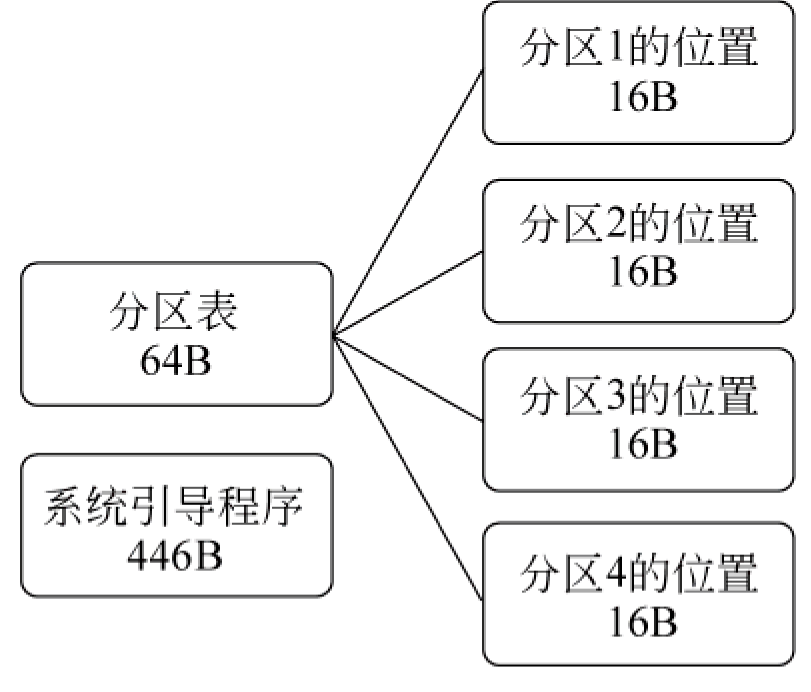


图 2.6　DPT 的结构

我们将记录在分区表中的分区称为主分区，可见一块硬盘最大可容纳 4 个主分区。当需要使用超过 4 个以上的分区结构时就需要对分区表进行扩充。那么该如何扩充呢？如图 2.7 所示。

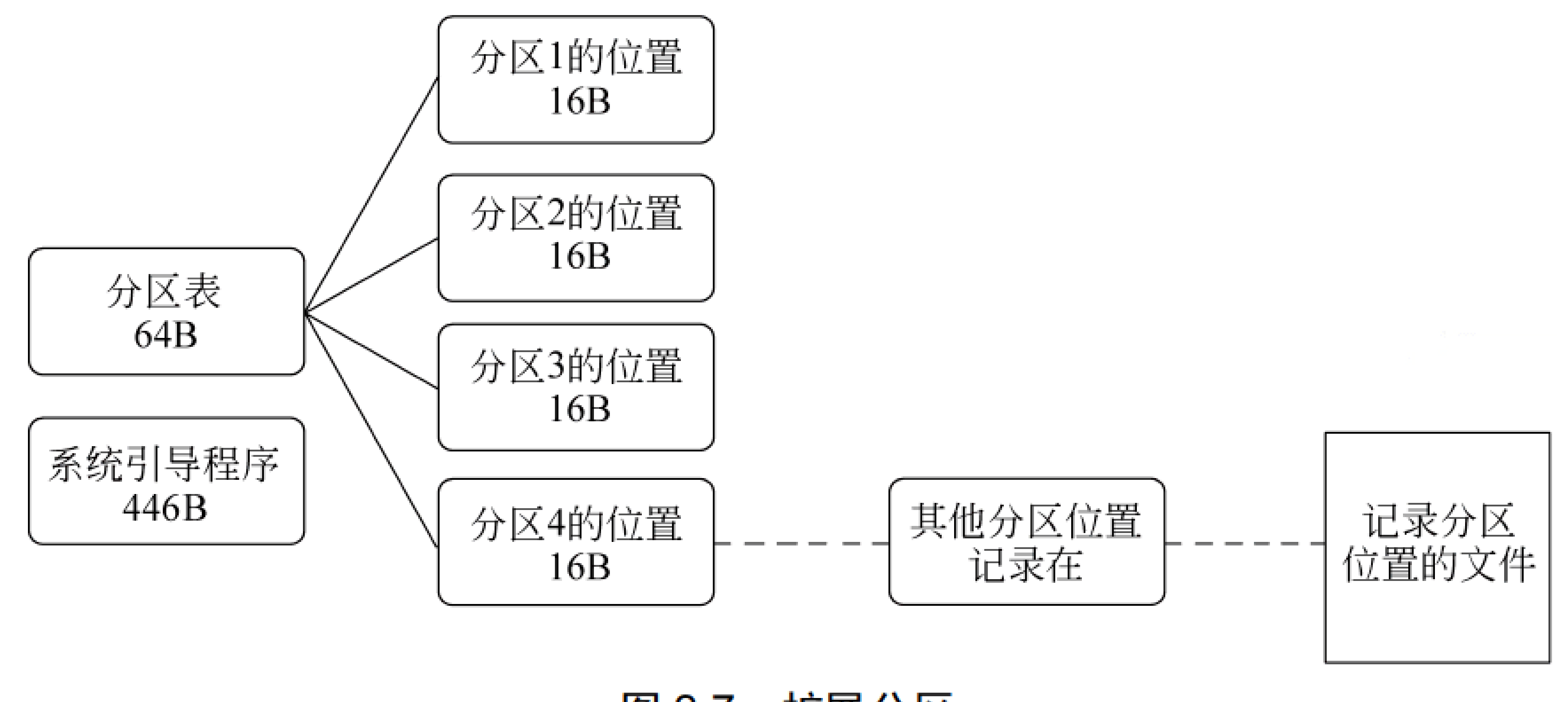


图 2.7　扩展分区

可以拿出分区表中一个记录磁盘分区位置的 16B 空间来记录一个位置，该位置实际上是一个文件。文件中清楚地记录了主分区以外的其他所有分区在磁盘中的位置，这个文件称为“扩展分区记录”。这样一来就可以突破 4 个分区的限制了。我们将记录“扩展分区记录”的那个磁盘分区位置称为扩展分区。在一个硬盘中只能有一个扩展分区存在。

记录在“扩展分区记录”中的分区称为逻辑分区。逻辑分区的数量在理论上是不受限制的，但是在实际的系统实现中，设备驱动程序会限制逻辑分区的数目，IDE 接口的硬盘不能多于 63 个逻辑分区，SCSI 接口的硬盘不能多于 15 个逻辑分区。

通过以上的论述，可以得出如下结论。

（1）一块硬盘中最大支持 4 个主分区。

（2）一块硬盘中仅支持一个扩展分区，且扩展分区要占用 1 个主分区的位置。

（3）逻辑分区的数量在理论上不受限制，但在具体的操作系统实现中被规定了数量的上限。

在了解了上述分区类型后，再回头看看图 2.5 所示的分区方案。基本上涵盖了分区的各种组合方式。

那么在 Linux 中分区是如何表示的呢？在 Linux 中使用“硬盘号+分区编号”的方法来表示（如图 2.8 所示）。

在/dev/hda1 这种表示方法中：

/dev/hda 是硬盘编号（硬盘设备文件）；

1 是分区的编号（第 1 个分区）。不论是 IDE 接口设备还是 SCSI 接口设备均采用这种编号方式。

/dev/hda1 表示的是/dev/hda 这块硬盘中的第一个分区；同理，/dev/hda2 表示的是/dev/hda 这块硬盘中的第二个分区，依此类推。

请看下例（如图 2.9 所示）：

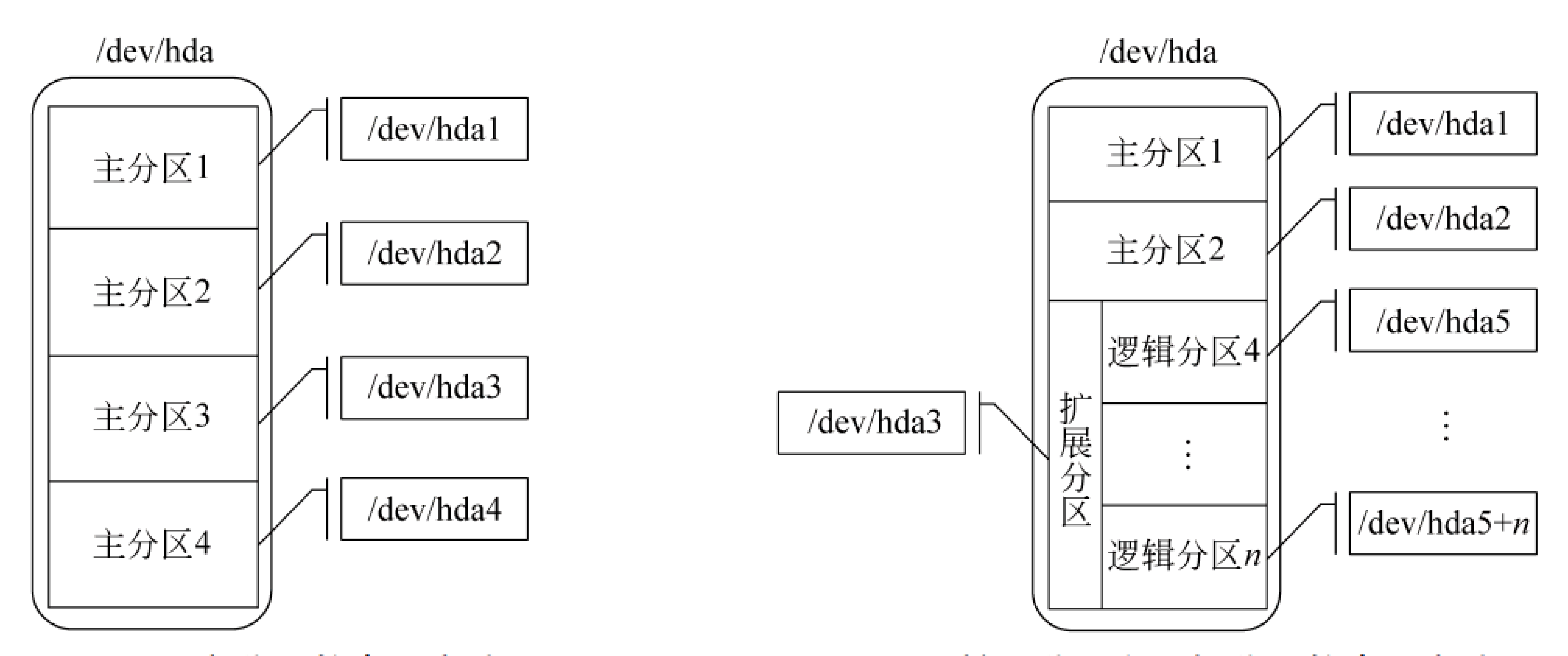


图 2.8　主分区的表示方法　　　图 2.9　扩展分区和逻辑分区的表示方法

在图 2.9 中包括两个主分区、一个扩展分区和若干个逻辑分区。主分区使用/dev/hda1 和/dev/hda2 表示，扩展分区因为是占用 1 个主分区位置，所以使用/dev/hda3 表示。需要注意的是，第一个逻辑分区使用/dev/hda5 来表示，而不论该分区在顺序上的排列位置为何。依此类推，第二个逻辑分区使用/dev/hda6 表示，等等。

2.2　CentOS Linux 的安装

对于 Linux 操作系统而言，其安装的方式可以分为以下 3 种。

（1）本地安装：利用安装光盘直接安装。即将安装光盘放入光驱，并将计算机调整到由光盘启动，从而直接启动安装光盘中的操作系统安装程序，实现操作系统的安装。

（2）网络安装：将操作系统的安装镜像文件存储于服务器中，本地主机使用 PXE 芯片实现网络启动并连接至服务器，下载安装镜像中的安装程序后在本机运行，实现操作系统的安装。

（3）自动安装：将操作系统的安装镜像文件存储于光盘或本地硬盘中，启动自动安装程序实现操作系统的自动安装。

不论哪种安装方式，在安装过程中都会遇到两种安装界面：图形安装界面和文字安

装界面。图形安装界面比较直观且支持中文，对初学者而言很容易掌握；文字安装界面的安装速度较快，适用于对 Linux 操作系统比较熟悉的用户或者不支持图形显示的计算机。

文字安装界面与图形安装界面的比较如表 2.4 所示。

表 2.4　不同安装界面的比较

比较内容	文字安装界面	图形安装界面
适用范围	所有待安装 Linux 的主机	支持显示 Linux 图形界面的主机
安装速度	快	慢
用户界面	字符界面，采用键盘操作	图形界面，可使用鼠标操作

对初学者而言，推荐图形安装界面，它有利于上手，且结构清晰明了。当掌握了图形安装界面后，对整个安装过程中的设置项就有了明确的认识了，这时再使用文字安装界面就很简单了，所涉及的仅仅是对安装选项的另一种配置方法。

2.2.1　CentOS 的图形界面安装

CentOS 5 的安装光盘有两种介质形式：CD 介质和 DVD 介质。CD 介质共有 7 张安装光盘， DVD 介质有 1 张安装光盘。考虑到 DVD 光驱已经发展为当前主机的标准配置了，下面采用 DVD 介质。

CentOS 5 的下载站点为：

32 位系统：http://isoredirect.centos.org/centos/5/isos/i386/

64 位系统：http://isoredirect.centos.org/centos/5/isos/x86_64/

下载文件为：

CD 介质：CentOS-5.5-i386-bin-1of7.iso
CentOS-5.5-i386-bin-2of7.iso
CentOS-5.5-i386-bin-3of7.iso
CentOS-5.5-i386-bin-4of7.iso
CentOS-5.5-i386-bin-5of7.iso
CentOS-5.5-i386-bin-6of7.iso
CentOS-5.5-i386-bin-7of7.iso

DVD 介质：CentOS-5.5-i386-bin-DVD.iso

在获得安装介质后，使用 CentOS 5 DVD 安装光盘引导系统，进入启动画面，如图 2.10 所示。这个界面是一个引导界面，可用来决定以什么方式开始 Linux 的安装或其他操作。

（1）To install or upgrade in graphical mode, press the <Enter> key. 即当需要使用图形界面安装或升级操作系统时，直接按回车键即可。

（2）To install or upgrade in text mode,type：linux text <Enter>.即当需要使用文字界面安装或升级操作系统时，需要（在最下方的“boot：”处）输入 linux text，然后按回车键。

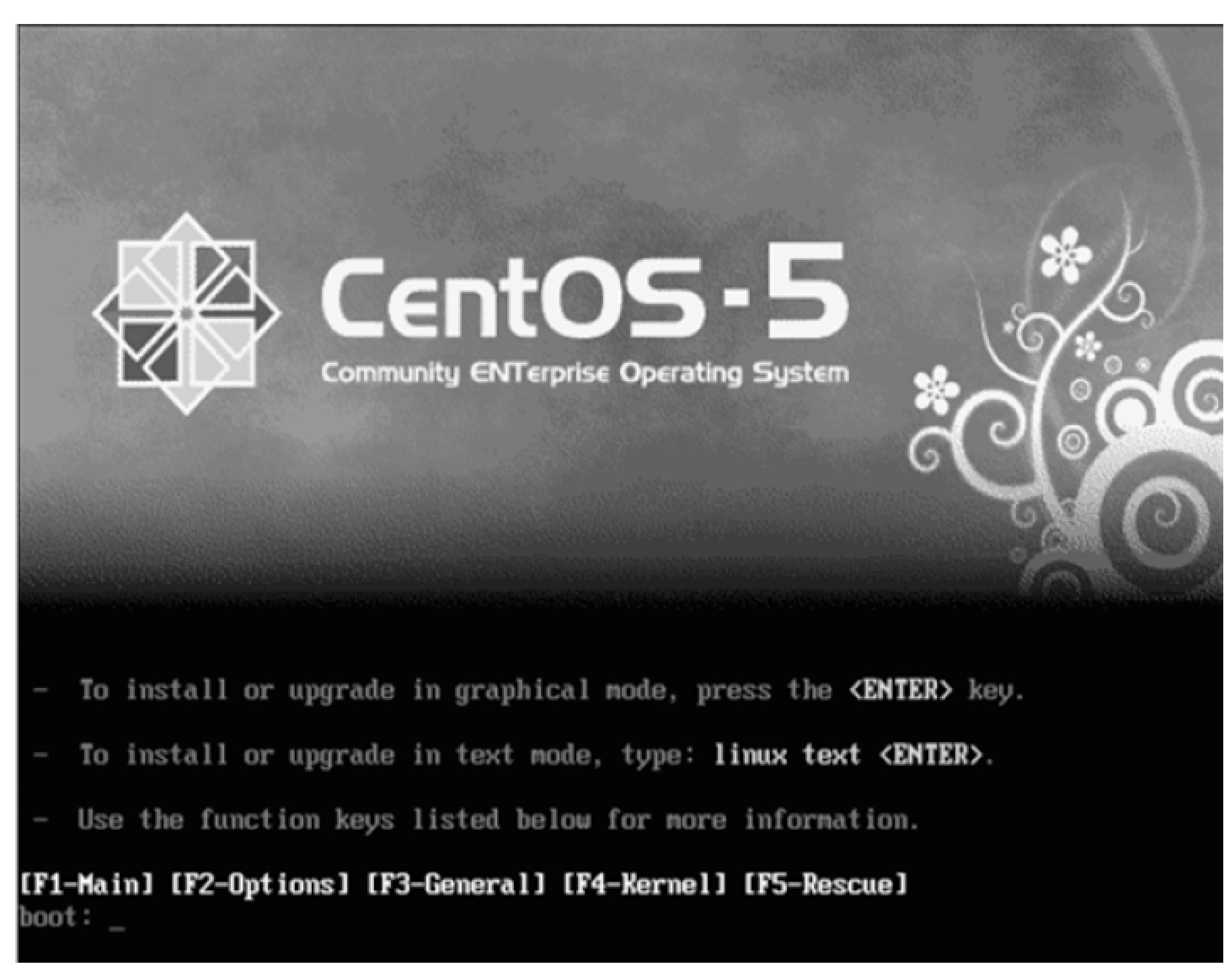


图 2.10　Linux 安装引导界面

（3）Use the function keys listed below for more information.使用功能键获得更多相关操作信息。这里的功能键是指最下方的 [F1-Main]、[F2-Options]、[F3-General]、[F4-Kernel]和 [F5-Rescue]。

[F1-Main]：F1 功能键将显示主菜单，即当前这个菜单界面。

[F2-Options]：F2 功能键将显示其他操作选项，常用的选项如下：

linux mediacheck：测试安装介质。

linux rescue：进入 Linux 的修复模式。

linux askmethod：提示用户选择本地安装还是网络安装，并决定采用何种网络安装方式。

[F3-General]：F3 功能键将给出一个操作提示。

[F4-Kernel]：F4 功能键允许用户在启动安装程序前添加内核参数。

[F5-Rescue]：F5 功能键将引导用户进入 Linux 的修复模式。

这里直接按回车键进入图形安装界面。

1. 测试安装介质

按回车键后，安装程序首先会询问用户是否进行安装介质的测试，如图 2.11 所示。

该测试可确定安装光盘介质自身的正确性，该功能的作用包括以下两方面。

（1）可以检测出光盘是否存在物理损坏。在第一次使用某套光盘进行安装时应先测试所有安装光盘是否完好，这样可以避免由于光盘的物理损坏而导致的安装失败。

（2）可以确保此光盘为官方发布版本，而没有经过任何人的篡改，从而提高了系统的安全性。

整个光盘的测试过程是由安装程序自动完成的，如图 2.12 所示。如果使用的是 CD 安装介质，则测试程序在检测完一张光盘后会自动弹出该光盘并提示用户更换光盘，直

到所有的安装光盘都被检测完成。



图 2.11　安装介质测试

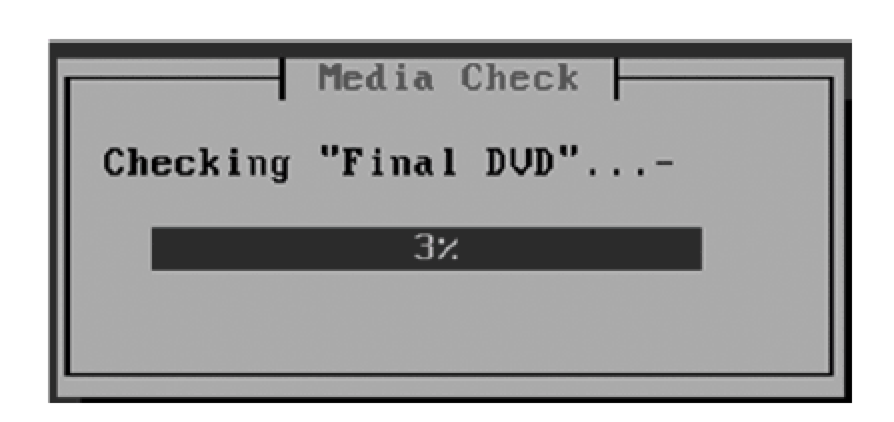


图 2.12　光盘测试过程

如果光盘介质可以正常使用，安装程序将提示用户可以使用该介质进行安装，如图 2.13 所示；否则，将要求用户更换安装介质。

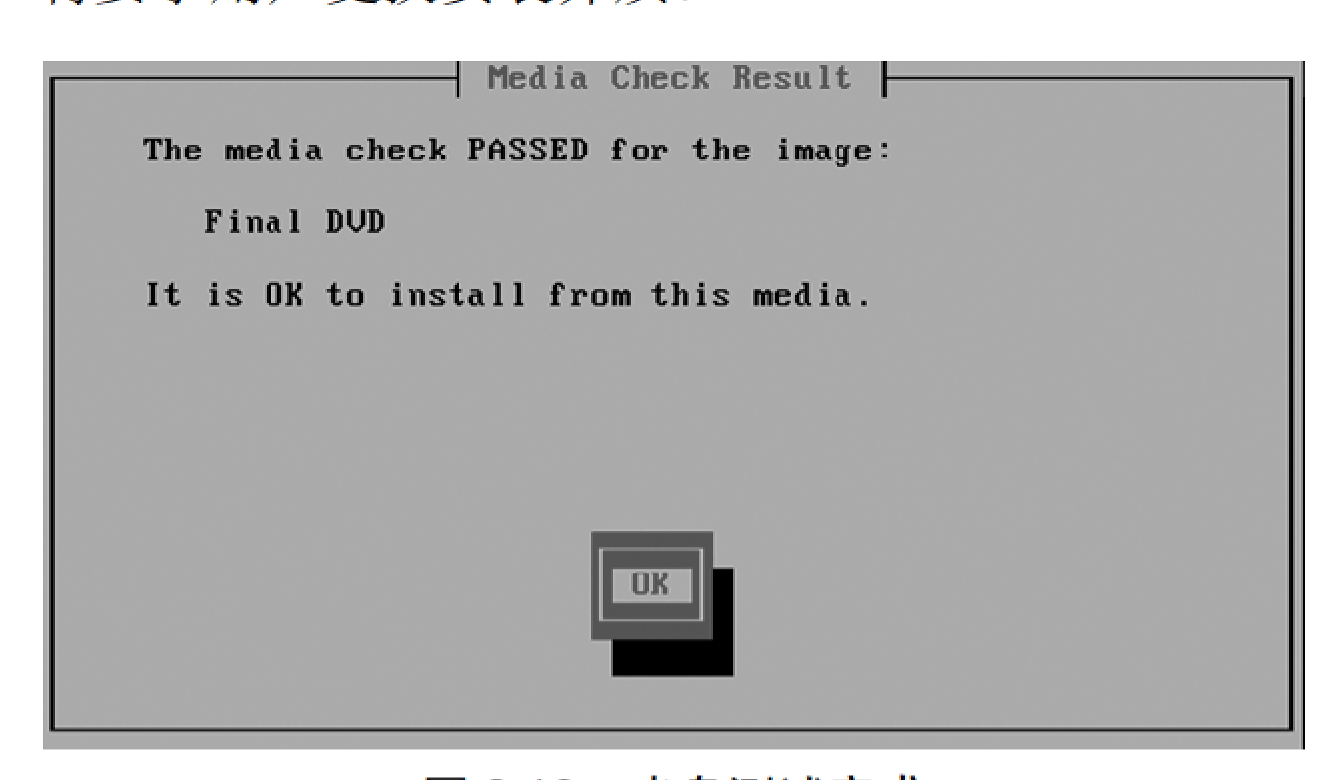


图 2.13　光盘测试完成

如果能确定光盘是完整可靠的，在图 2.11 所示的界面中可选择 Skip 跳过测试光盘介质的步骤（使用 Tab 键在 OK 和 Skip 两个选项之间切换）。

接下来，安装程序会启动图形安装界面，开始操作系统的安装设定了。

2. 选择安装界面语言

进入图形安装程序后，首先看到的是一个欢迎界面，如图 2.14 所示，在该欢迎界面中单击 Next 按钮开始下一步安装，选择安装界面使用的语言。

CentOS 的安装程序支持多种语言界面，包括英文、简体中文和繁体中文等。这里可以选择简体中文，如图 2.15 所示。这时整个安装过程将采用中文进行，系统安装完成后也将采用简体中文作为默认的语言环境。单击“下一步”按钮进入下一步安装。

3. 选择键盘类型

CentOS 的安装程序会自动检测当前计算机所使用的键盘的类型，这个检测值是可信的。通常国内普遍使用的标准的 101 键盘被识别为“us，美国英语式键盘”，可以直接使用该检测值，如图 2.16 所示。单击“下一步”按钮进入下一步安装。



图 2.14　安装程序的欢迎界面



图 2.15　选择安装界面语言环境



图 2.16　选择键盘布局

4. 初始化硬盘

接下来要定义的是将操作系统安装到哪个磁盘分区的问题。如果是在一块新的硬盘上安装操作系统，则安装程序会提示用户因无法读取硬盘的分区表，需要对磁盘进行初始化。这时，需要同意（单击“是”按钮）该操作才能继续安装，如图 2.17 所示。如果是一块使用过的硬盘则不会遇到该提示。

5. 选择分区方案

现在开始为安装准备相应的磁盘分区，此处假设用户在一台新的机器上安装 CentOS，硬盘中没有任何需要保留的数据，可以全部用于安装 Linux。

CentOS Linux 提供了 4 种分区方案供用户选择，如图 2.18 所示。

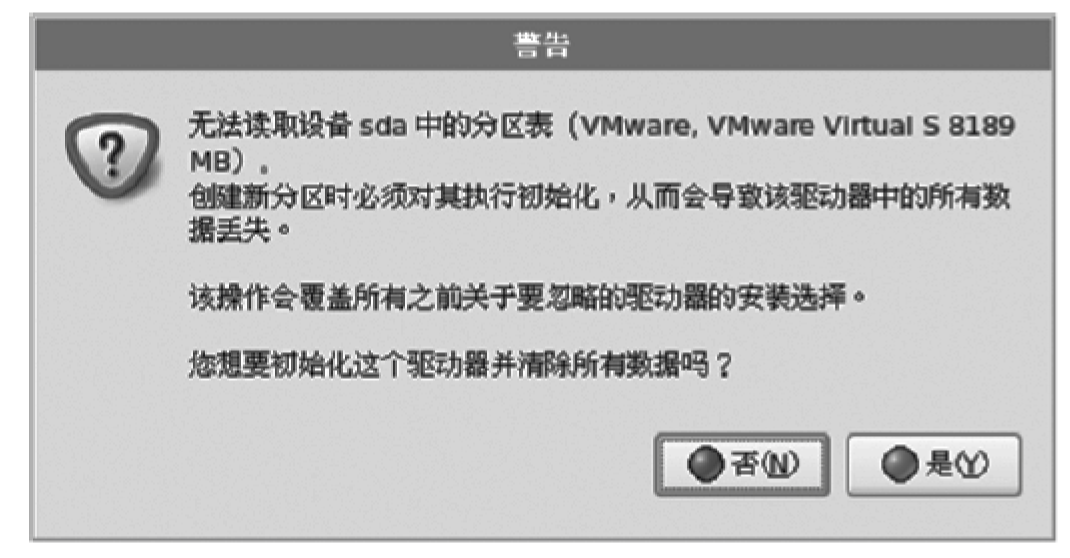


图 2.17　初始化硬盘提示



图 2.18　CentOS 提供的分区方案

（1）在选定磁盘上删除所有分区并创建默认分区结构：该选项将删除磁盘上的所有分区，将整个磁盘用于 Linux 操作系统的分区。

（2）在选定驱动器上删除 Linux 分区并创建默认的分区结构：该选项将删除磁盘上已有的所有 Linux 分区，并将这些被删除的 Linux 分区重新整合为用于 Linux 操作系统的分区。

（3）使用选定驱动器中的空余空间并创建默认的分区结构：该选项将使用磁盘上的空闲空间作为 Linux 操作系统的分区。

（4）建立自定义的分区结构：该选项将要求用户自行建立 Linux 操作系统的分区结构。

为了更准确地控制磁盘空间的分配，这里选择“建立自定义的分区结构”，如图 2.19 所示。



图 2.19　建立自定义的分区结构

选择“建立自定义的分区结构”后，安装程序将要求用户定义分区。这里先创建两个分区，分别为根分区/和交换分区 swap。

根分区/用于包含整个 Linux 的文件系统，交换分区 swap 用于形成 Linux 的虚拟内存结构。Linux 的分区种类很多，可以单独定义的分区也很多。在初次接触 Linux 时可以先定义这两个分区结构，至于更精确的分区结构可以在后续课程中逐渐地接触与添加，这样对于分区结构这个问题会有一个更明确的认识。

（1）新建分区。选择磁盘驱动器，并选择“新建”选项定义分区，如图 2.20 所示。

（2）建立根分区“/”。如图 2.21 所示，建立根分区的方法是挂载点为/，文件系统为 ext3，分区大小根据硬盘的实际情况和待安装软件包的情况酌量安排。本例中整个硬盘



图 2.20 新建磁盘分区

的大小为 8GB，定义根分区的（/）的容量为 7.5GB。图 2.21 中“固定大小”是指分区和指定的大小完全相同。



图 2.21 建立根分区

（3）建立交换分区 swap。与创建根分区的操作过程相同，首先选中磁盘中的空闲空间，并选择“新建”选项定义分区，如图 2.22 所示。

建立交换分区的方法是挂载点为空，即挂载点选项保持空值，文件系统为 swap，分区大小根据内存的实际情况而定，应为物理内存容量的两倍。

至此，为 Linux 操作系统定义了两个分区：根分区和交换分区（如图 2.23 所示）。

由图 2.23 中可知，当前是一块 SCSI 接口的硬盘，该硬盘是 SCSI 接口卡上的第一个 SCSI 设备（/dev/sda）。在该硬盘上划分了两个分区，/dev/sda1 为根分区“/”，/dev/sda2 为 swap 交换分区。单击“下一步”按钮继续进行安装。

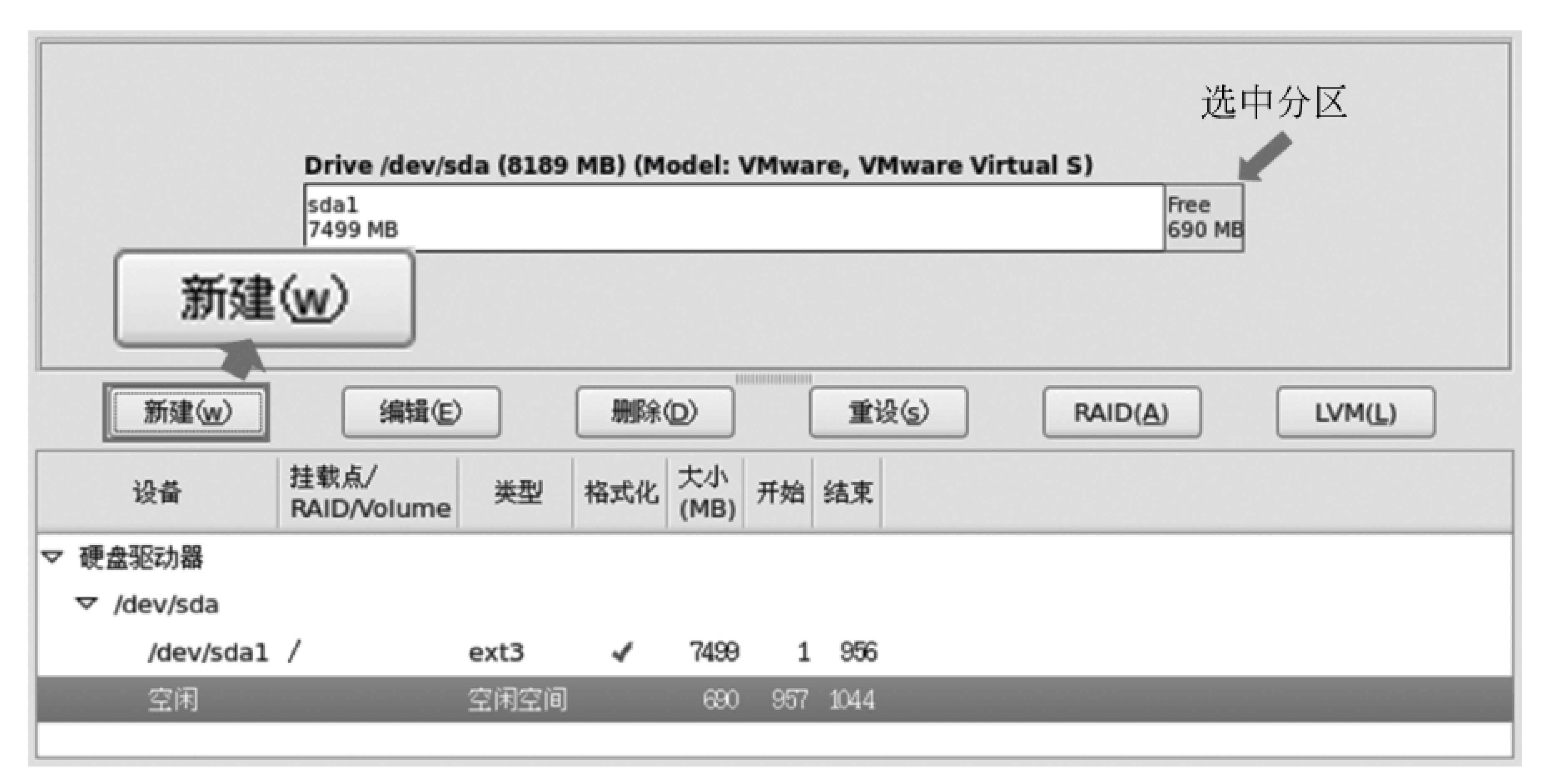


图 2.22　选中磁盘的空闲空间

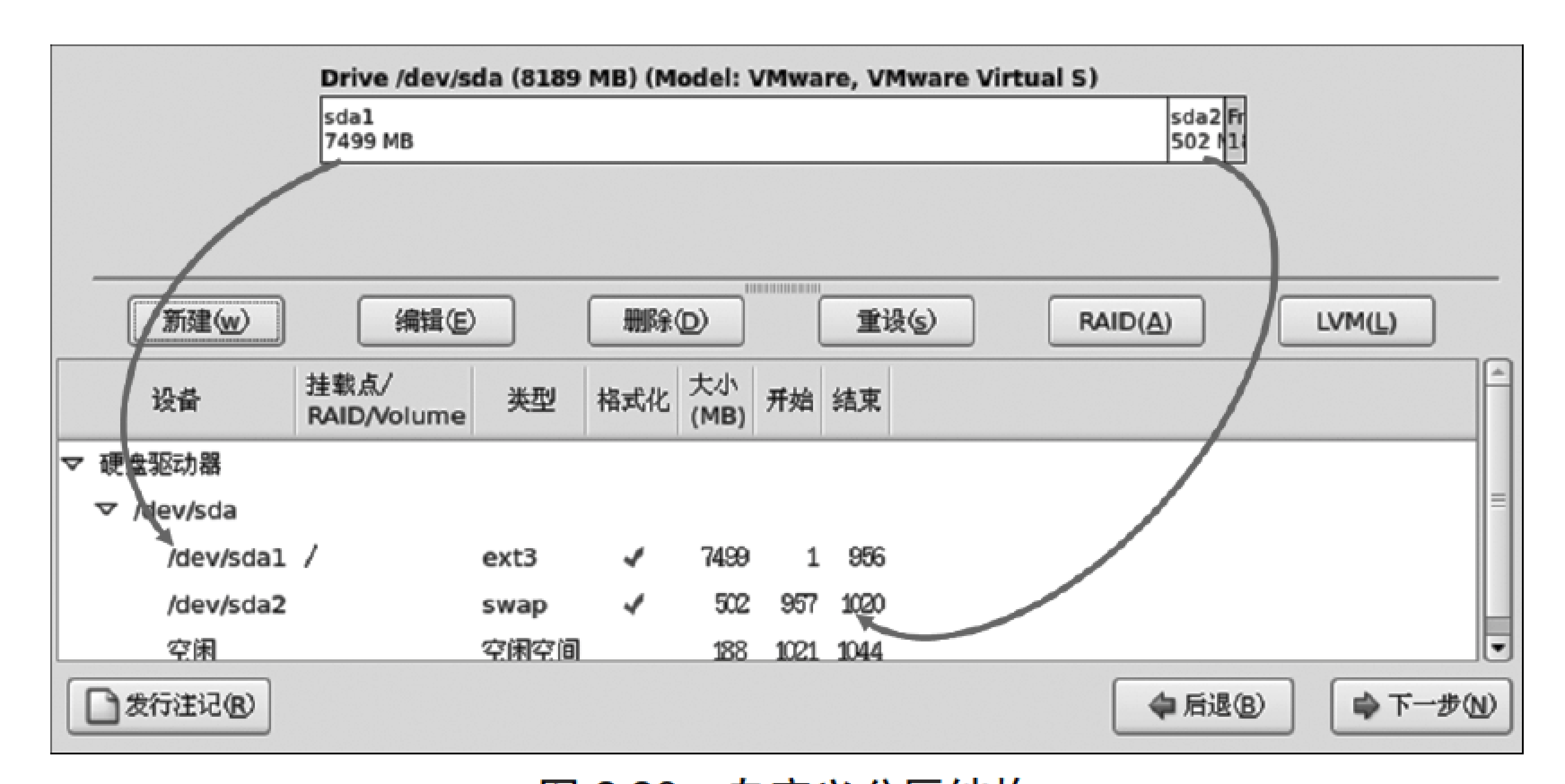


图 2.23　自定义分区结构

6. 启动引导器设置

启动引导器的设置对于引导已安装的操作系统正常启动是至关重要的，对于 Linux 而言，常见的有两种启动引导器可供选择：LILO 和 GRUB。有关这两种启动引导器的详细内容后面会专门进行讨论，这里先进行设置，以便能正常引导 Linux。

在 CentOS 中默认仅提供 GRUB 启动引导器供用户使用，而不再提供 LILO 启动引导程序了，其原因在于 GRUB 的功能比 LILO 要强大的多。

如图 2.24 所示，选择 GRUB 引导装载程序将会被安装到/dev/sda 上，这样 GRUB 就可以引导 Linux 启动。如果当前计算机中还存在其他操作系统，GRUB 会自动检测到操作系统信息，并将检测结果显示在启动标签位置供用户选择默认启动哪个操作系统。也就是说，GRUB 不仅可以引导 Linux 操作系统，也可以引导 Windows 等其他操作系统。

单击“下一步”按钮，进入网络接口设置界面。

7. 网络接口设置

由于 Linux 操作系统通常作为网络服务器使用，所以其网络配置就显得格外重要，

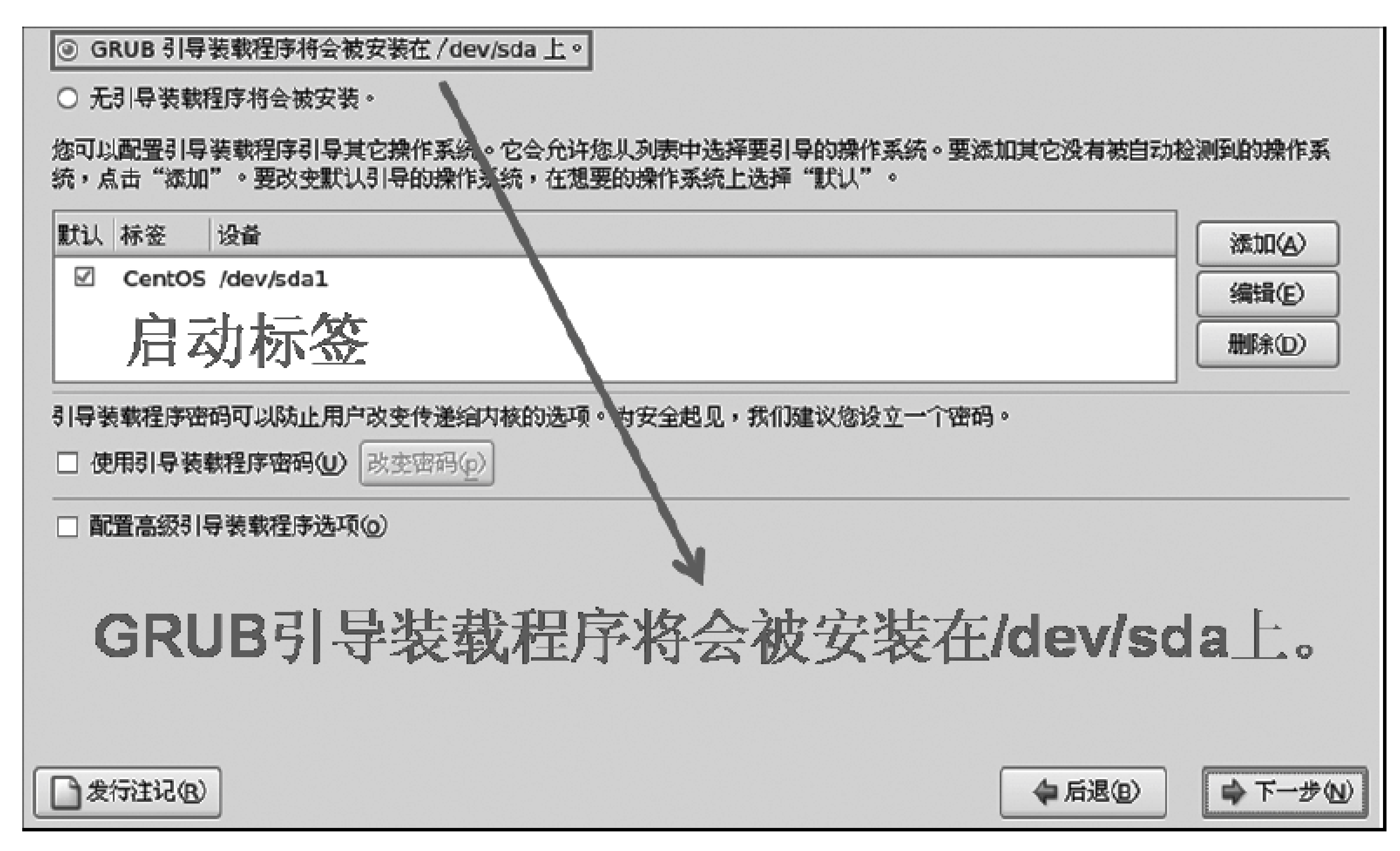


图 2.24　安装 GRUB

在 CentOS 的安装过程中提供了基本的网络配置步骤，使得系统启动后能够进行正常的网络连接。对于网络的进一步配置将在后续章节中详细介绍。

1）网卡设置

CentOS 的安装程序能够自动检测出主机中所使用的网卡，并给出配置界面，如图 2.25 所示。



图 2.25　定义网络连接

从图 2.25 中可见，主机中的网卡被安装程序自动检测到，命名为 eth0 并采用 DHCP 的方式分配 IP 地址等相关网络连接信息。当需要自定义网络连接信息时可以单击“编辑”选项，以决定是否启用 DHCP、是否启用 IPv6 支持、主机的 IPv4 与 IPv6 地址信息等。有关如何设置网络的 IP 地址、子网掩码等信息，需要咨询本地网络管理员。相关的知识结构，包括后面的网关和 DNS 配置将在第 17 章详细介绍。

2）主机名设置

对于主机名信息为默认的情况下也是通过 DHCP 服务器自动获得的。如果需要自定

义主机名，则需要选择“手工设置”单选按钮，如图 2.26 所示。

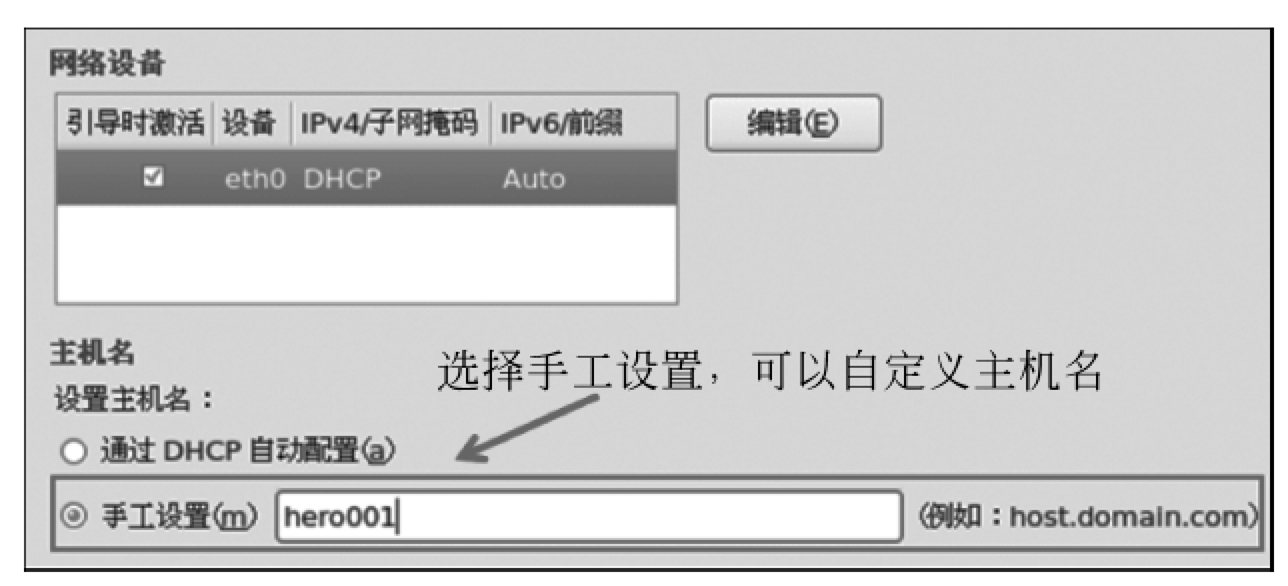


图 2.26 手工设置主机名

此处选择手工设置，用户可自定义主机名。同时，如果网卡未使用 DHCP 自动获取 IP 地址，则此处还可以设置网关地址以及 DNS 服务器地址的网络连接信息。单击“下一步”按钮继续安装过程。

8. 设置时区

在该步骤中设置机器的时区信息，如图 2.27 所示。时区信息中包括本地时间协议，比如夏令时。另外 Linux 允许将 BIOS 时钟设定为本地时间，也可以设定为全球时间（UTC）。后者是更方便的方法，特别是对需要经常调整时区的笔记本计算机用户。当改变时区时，BIOS 根本不用调整，只要提供可用来辨析时区的信息就可以了。但是，其他操作系统要求将 BIOS 时钟设为本地时间，所以这个选项对于多操作系统并存的环境并不适合。

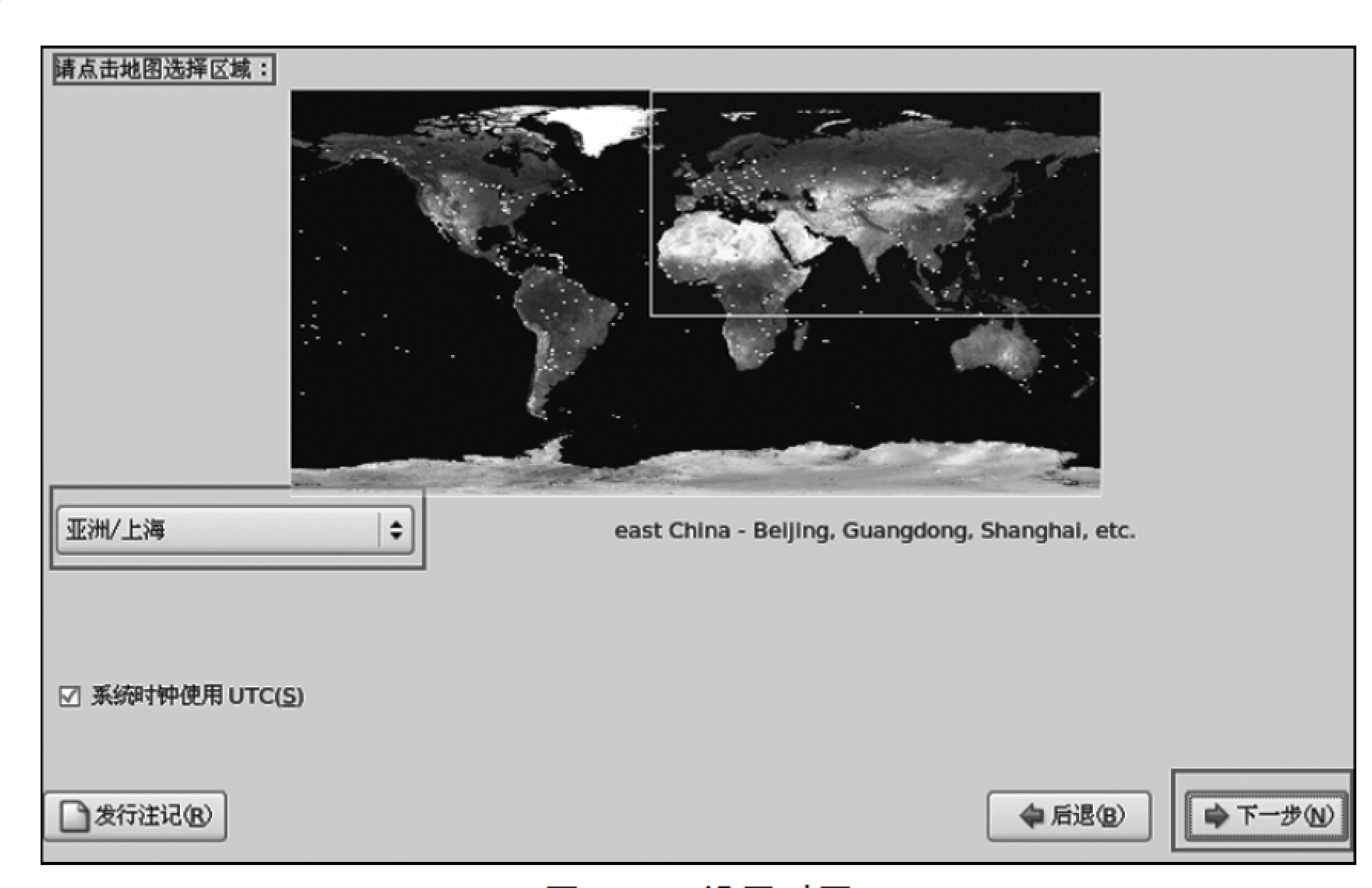


图 2.27 设置时区

9. 设置管理员口令

Linux 操作系统的管理员用户（根用户）默认为 root。该用户对系统有完全的控制权

限，所以该用户的登录密码应使用尽量不容易被破解的复杂密码，以提高该用户的安全性。密码的设置方法如图 2.28 所示，分别在根密码和确认位置输入两次同样的密码即可。



图 2.28　设置管理员密码

注意根密码和确认的输入必须一致，否则将提示“输入的口令不匹配，请重新输入”。

刚才提到了复杂密码的问题，那么什么样的密码才能算作是复杂密码呢？一个密码如果能满足下列的 4 个条件，则该密码就可以认为是一种复杂密码了。当然，这不是硬性规定而是一个建议。

（1）密码的最小长度在 8 位以上。

（2）密码应该是大写字母、小写字母、数字和特殊符号（如#、$、%和&等）这 4 种符号中的 3 种以上的组合。

（3）密码不应包含个人信息。

（4）密码应容易记忆。

10. 选择需安装的软件

安装程序接下来会查找并决定安装什么样的软件。安装程序首先会询问是要接受默认选项，还是要自定义选择软件，如图 2.29 所示。

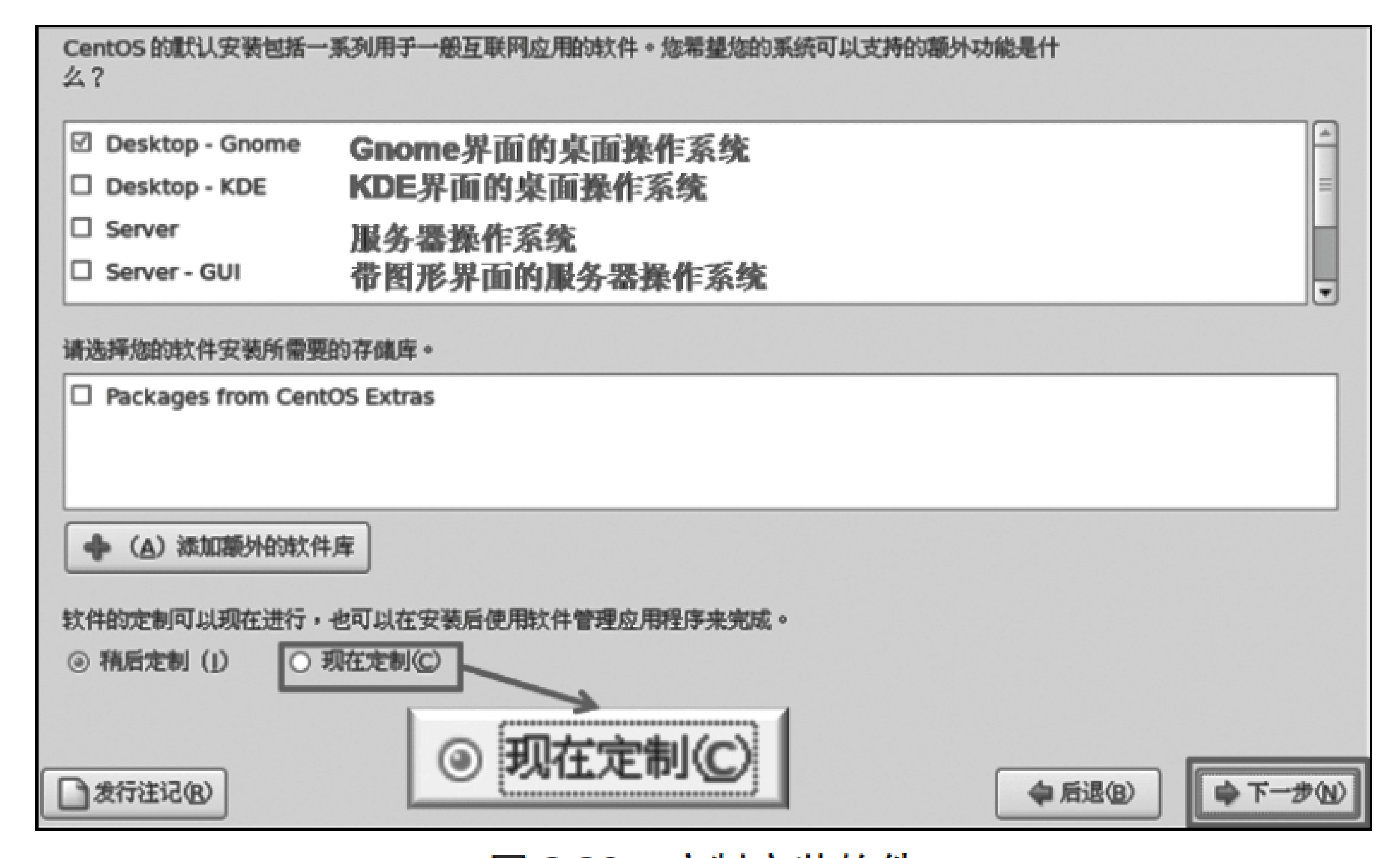


图 2.29　定制安装软件

如果要自定义软件安装，则选择“现在定制”，安装程序接下来会显示软件组件的选择页面，如图 2.30 所示。

软件分类组又进一步细分为小的软件组，包含多个可选软件组分类。每个软件组又分为必须安装的软件包和可选的软件包。如果选择浏览组件组的可选软件包，可以选定或取消特定软件包的安装，如图 2.31 所示。

可供安装的软件包的数目多、范围广，偶尔漏装个别软件包的妨碍不大，因为以后可以在系统中方便地添加或删除软件。

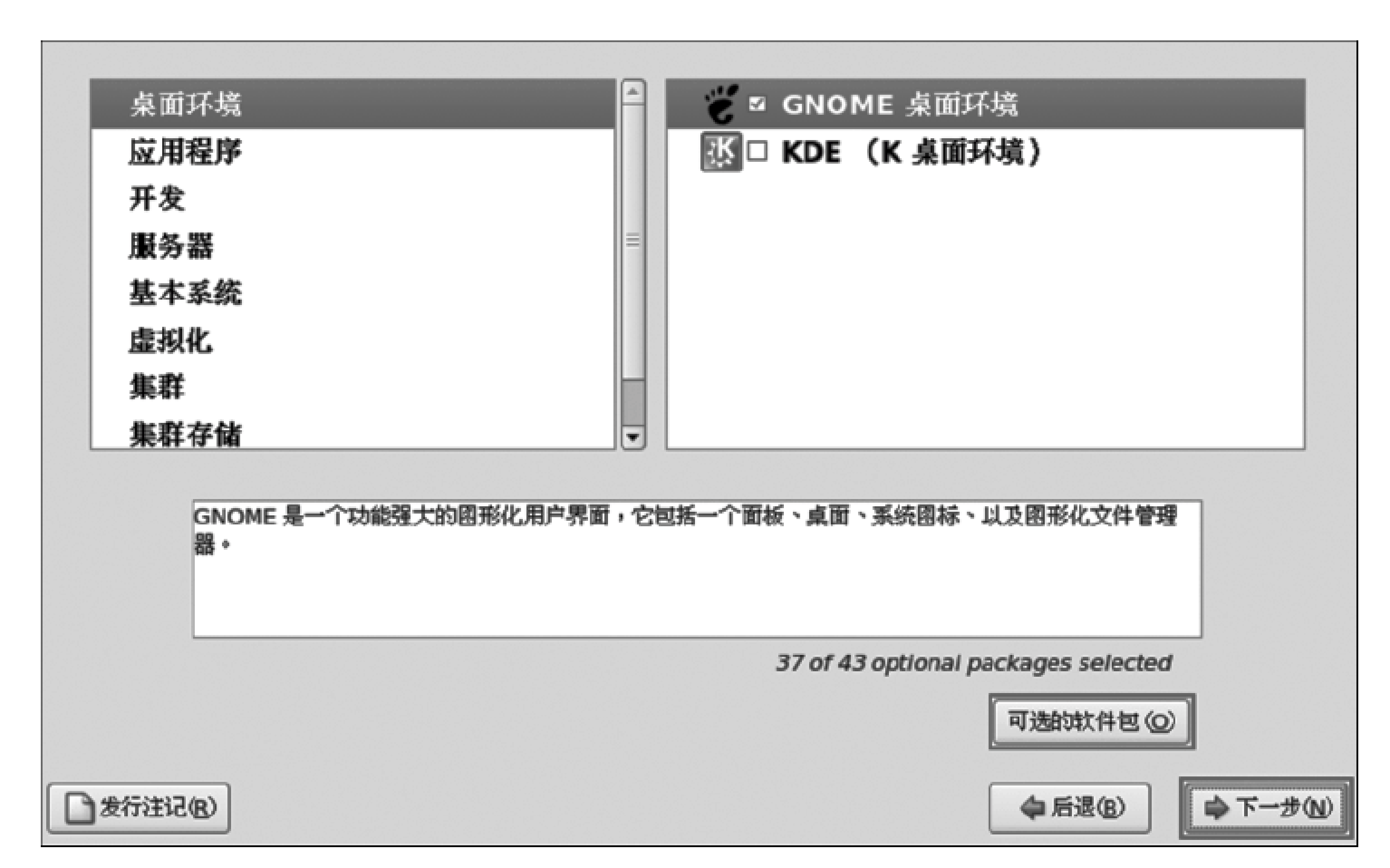


图 2.30　软件分类选择界面



图 2.31　软件自定义界面

11. 确认安装

经过上述步骤后即可开始软件的安装，首先安装程序将向用户显示一个提示信息，如图 2.32 所示。

单击“下一步”按钮即开始安装过程，如果使用的是 CD 安装光盘，这时系统将提示需要准备 5 张安装光盘，并且需要注意安装进度以便随时替换安装光盘。由于本书采用的是 DVD 安装光盘，所以不需要更换安装光盘，接下来的工作就是由安装程序自行完成。



图 2.32　确认安装

当安装完成后，安装程序将出现“重新引导”的提示，如图 2.33 所示。选择“重新引导”，系统将被重新启动。

2.2.2 操作系统的初始配置

重新引导计算机后，需要完成对计算机的初始配置。首先出现如图 2.34 所示的“欢迎”界面。单击“前进”按钮开始进行初始化配置。



图 2.33　安装完成



图 2.34　“欢迎”界面

CentOS 为强化系统安全内置了系统防火墙功能，并默认启用该功能。出于学习和实验的目的，此处先关闭防火墙功能（禁用该功能），如图 2.35 所示。以后会详细介绍防火墙的配置方法。

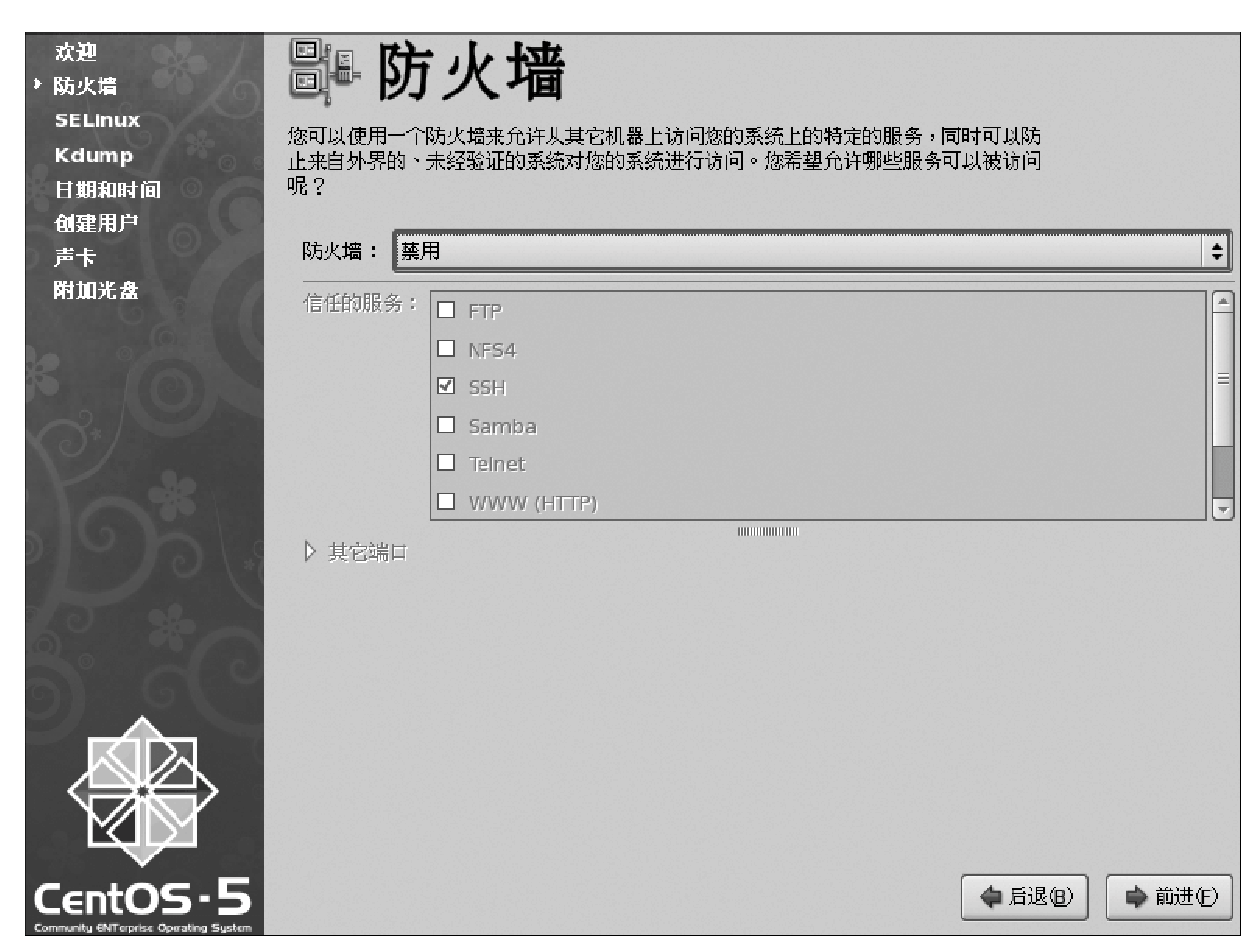


图 2.35　防火墙配置

禁用防火墙功能之后，系统会提示此设置会覆盖默认配置，单击“是”按钮，如图 2.36 所示，继续操作。

SELinux 是 CentOS 提供的另一种安全机制，同样是考虑学习环境的配置，这里也要暂时禁用 SELinux 功能，待了解该功能后再进行配置，如图 2.37 所示。

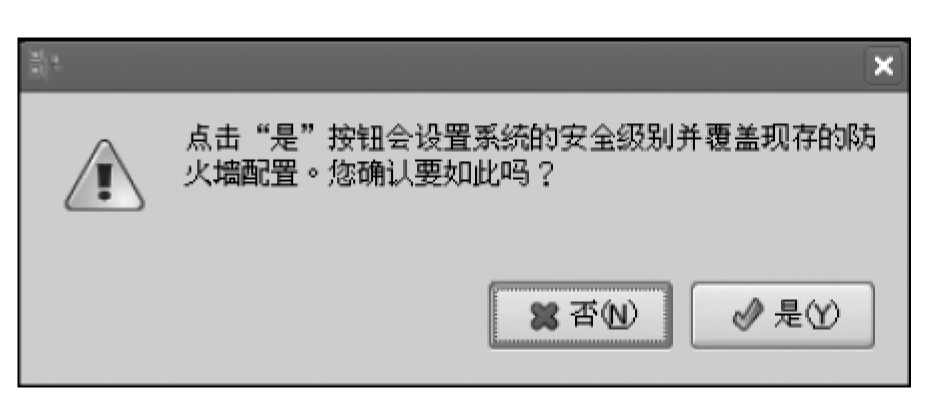


图 2.36　确认防火墙配置

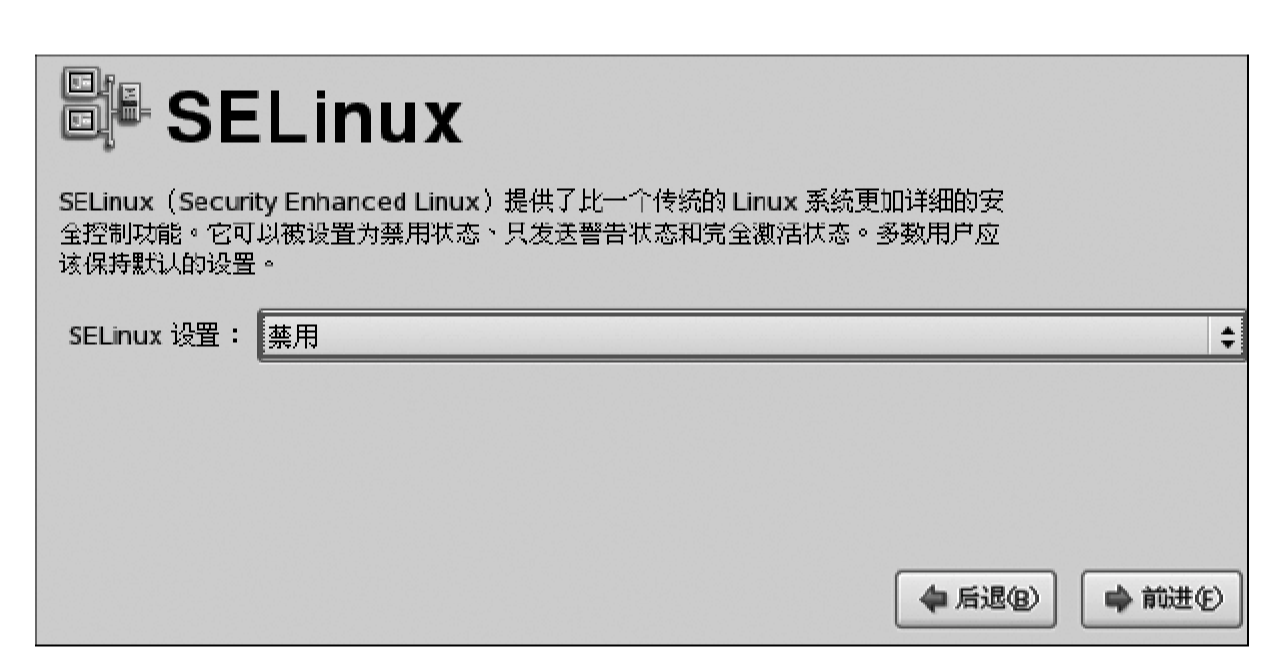


图 2.37　禁用 SELinux 功能

禁用 SELinux 之后，系统会提示用户该操作需要重新启动操作系统。单击“是”按钮继续操作，如图 2.38 所示。在完成整个配置之后系统将会重新启动。

在配置完 SELinux 后，配置向导会提示用户是否启用 Kdump，如图 2.39 所示。此项功能用于在当前系统内核发生故障时切换到另一个内核，在这个内核的支持下可以对崩溃的系统进行信息收集与调试。

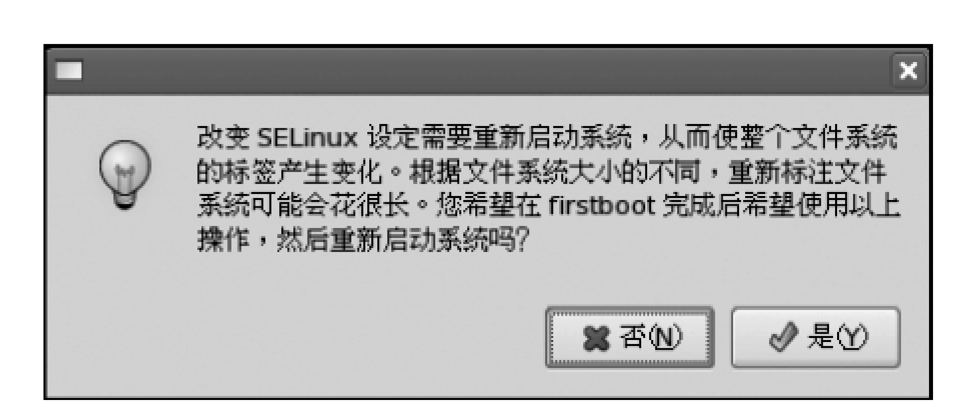


图 2.38　确认禁用 SELinux



图 2.39　配置 Kdump 服务

单击“前进”按钮，进入系统时钟的配置，如图 2.40 所示。



图 2.40　配置系统的日期和时间

系统的日期和时间来源于 BIOS 的时钟信息，若此处时间不正确，则需要进行相应的调整。单击“前进”按钮进入“创建用户”设置，如图 2.41 所示。

图 2.41　创建登录用户

Linux 推荐以普通用户身份登录系统而不要以管理员 root 用户登录，这是出于系统安全考虑的一种建议。此处允许用户创建管理员 root 以外的用户。这里可以不创建用户，具体的用户管理与规划可以在登录系统后统一完成。

声卡测试用于验证和配置系统声卡，如图 2.42 所示。通过该界面可测试系统对声卡是否正确识别。若对声卡的识别有问题，可以在登录系统后手工调整相应的驱动程序来加载声卡。

图 2.42　声卡测试程序

在配置程序的最后，配置向导会询问用户是否要安装附加软件，如图 2.43 所示。若要安装附加软件可以在此处进行。



图 2.43　添加附加软件资源

最后，单击“完成”按钮就结束了系统的初始化配置。由于在前面的步骤中禁用了 SELinux 的功能，所以系统会重新启动。

至此就完成了 Linux 操作系统的安装。

第3章　Linux 的基本操作

从本章开始将对 Linux 操作系统的管理性内容进行学习。Linux 操作系统有其特有的概念和组织方式，如果读者之前有过其他操作系统（如 Windows 操作系统）的使用经验，那么从现在开始应尽快适应 Linux 的概念和操作方式。随着学习内容的深入，读者会逐渐感受到 Linux 独特的魅力。

本章先由图形界面的登录和使用开始，介绍图形界面的基本使用、终端界面的基本结构、联机帮助和关机等主题，目的是帮助大家尽快了解 Linux 操作系统的使用方式，以便尽快上手实践。

使用 Linux 操作系统，首先要启动用户与操作系统之间的会话，所谓会话就是使用一种界面来操作 Linux 操作系统。

Linux 有着 UNIX 操作系统的特点与传统，早在 UNIX 操作系统发展初期，计算机使用的是一种“主机-终端”的硬件工作模型，如图 3.1 所示。这种工作模型类似于今天的计算机网络，但还不是严格意义上的计算机网络。当时，计算机是以主机为中心，多个用户通过终端连接至主机的串行接口上，实现对主机资源的使用。

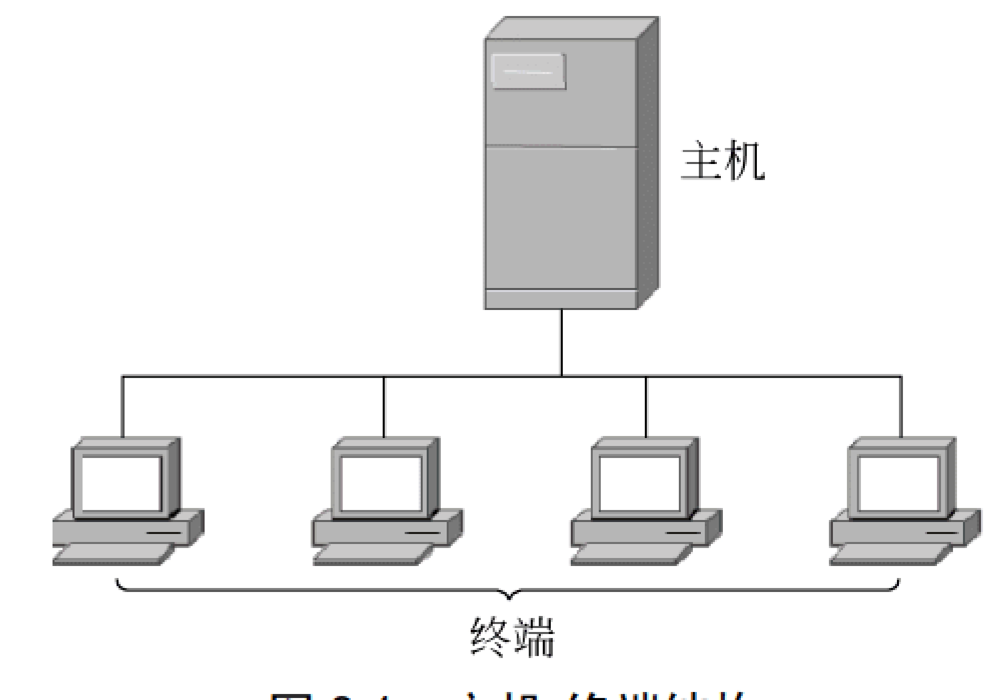


图 3.1　主机-终端结构

主机负责接收用户命令，解释和执行用户命令，将用户命令的执行结果返回给用户，是整个计算活动的中心节点。主机没有输入输出设备，所需的输入输出操作需要依靠终端设备完成。一台主机可以配置多个终端，以实现多用户使用。

终端负责利用键盘将用户的命令传递给主机，并将主机的运算结果利用显示器返回给用户，是整个计算活动的输入输出接口。对于终端而言，实际上就是键盘和显示器两个设备。

主机-终端结构的工作模式如图 3.2 所示。

主机-终端的这种工作模型实现的是一种以主机为中心的集中式运算结构，这需要主机操作系统能够支持多用户的应用环境。而作为传统主机操作系统的 UNIX 本身是一种多任务、多用户操作系统。它允许多个用户同时登录系统，使用系统资源。不同的用户使用不同的终端，利用各自的用户 ID（userid）和密码（password）向主机验证身份，登录系统。

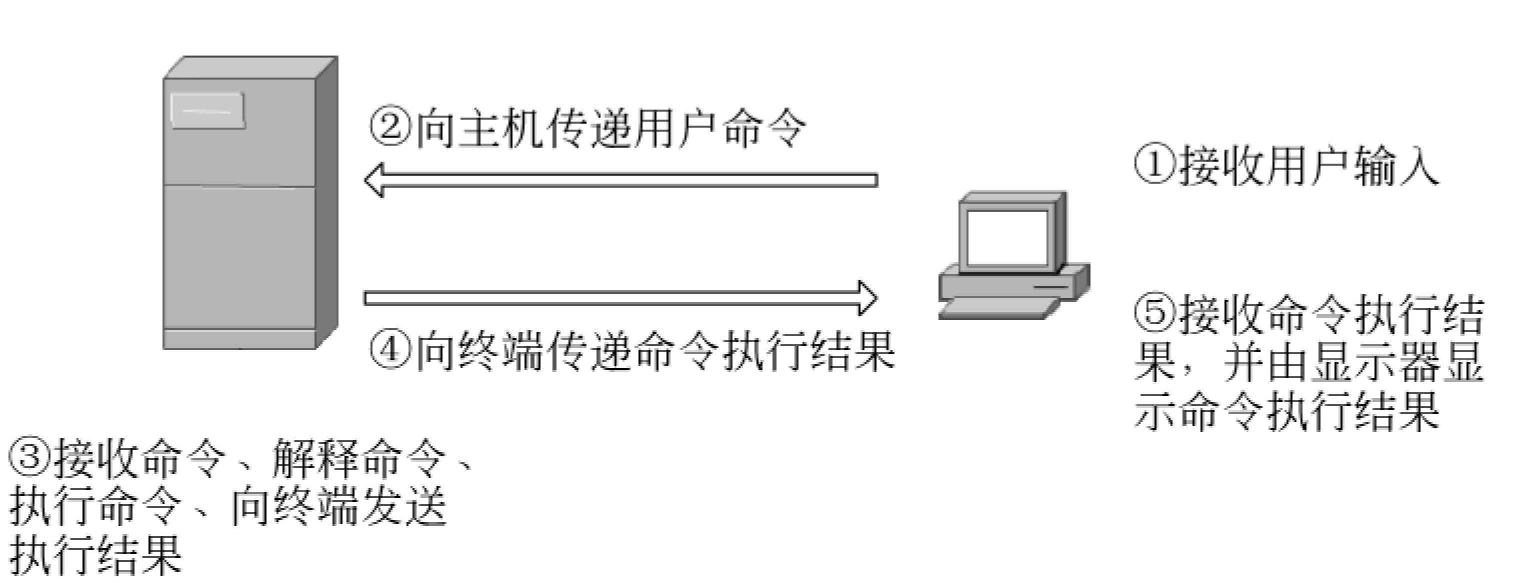


图 3.2　主机-终端的工作模式

尽管当前 PC 实现了主机的输入输出本地化结构，但 Linux 还是保持了 UNIX 的这一传统的工作模式。Linux 默认提供了 6 个虚拟控制台（Virtual Console），用 tty1~tty6 来表示；同时，Linux 的图形终端用 tty7 来表示。Virtual Console 又称为虚拟终端、文字终端或文字模式。

Linux 除虚拟终端这种以命令行操作为主的会话方式外，还提供了图形界面的会话方式。尽管图形界面的使用不是本书的重点，但是由于我们可能会在这个环境下使用一些应用程序，所以在全面介绍虚拟终端之前需要对图形界面作简要的介绍，对于图形界面的更多的内容请参考其他资料。

3.1　使用图形界面登录系统

在成功地安装完操作系统后，就可以登录该系统来体验这个全新的系统环境了。按照系统认知的顺序，首先向大家介绍在图形界面下如何登录系统、图形环境的基本操作以及如何注销和关闭系统。

启动 CentOS 5.5 后，将看到该系统的图形登录界面。在该界面中有 3 个地方需要注意，如图 3.3 所示。



图 3.3　图形登录界面

① 为显示的当前系统时间和主机名；

② 为用户验证区，在此处输入用户名和密码进行身份验证；

③ 为启动功能区，用于设置系统启动语言、会话、重新启动系统以及关机操作。

先来看启动功能区，该区域共有 4 组菜单项："语言"、"会话"、"重新启动"和"关机"。

"语言"菜单项用于定义用户登录系统后所使用的语言环境，即以什么语言完成显示、存储等操作，如图 3.4 所示。默认的情况下，系统使用操作系统安装时选择的语言环境。当需要更改系统使用的语言环境时，单击"语言"菜单，在弹出的语言菜单列表中选择需要的语言之后，单击"更改语言"按钮即可。

"会话"菜单项用于定义用户登录系统后所使用的会话。所谓"会话"是指系统为用户提供的操作环境，包括是否支持鼠标操作、菜单的排列方式、默认提供的系统工具种类等一些列与用户使用操作系统相关的环境。CentOS Linux 提供了 3 种会话供用户选择，分别为 GNOME、KDE 和安全模式终端，如图 3.5 所示。



图 3.4 选择会话语言环境



图 3.5 选择会话环境

（1）GNOME：即 GNU 网络对象模型环境（GNU Network Object Model Environment），是 GNU 计划的一部分，也是开放源码运动的一个重要组成部分。它是一种让使用者容易操作和设定计算机环境的工具。GNOME 的目标是基于自由软件，为 UNIX 或者类 UNIX 操作系统构造一个功能完善、操作简单以及界面友好的桌面环境，它是 GNU 计划的正式桌面，也是 Linux 系统默认使用的会话。

（2）KDE：是一个用于 UNIX/Linux 工作站的网络透明的现代化桌面环境，类似于 MacOS 和微软的 Windows 的桌面环境。与 GNOME 一样，KDE 同样提供了一个功能丰富、易于使用的图形应用界面。

（3）安全模式终端：是一个命令行界面，仅支持使用命令对系统进行控制。由于启动该界面时不会加载预先的启动设置，所以该会话经常在系统故障排除中被使用。

我们先以 GNOME 会话模式登录，来熟悉一下整个图形界面。

在用户验证区输入用户名 root 与密码。root 是系统的管理员用户名，密码是在安装系统时定义的密码。登录系统后，将载入 GNOME 图形界面。

注意，如果更改过语言或会话选项，则系统会询问用户此项修改是针对本次登录的临时修改，还是作为系统默认的永久修改。用户可根据需要在两者之间选择。

3.1.1 GNOME 图形界面介绍

成功登录 GNOME 会话环境后，首先看到的就是 GNOME 的桌面，如图 3.6 所示。



图 3.6 GNOME 界面

先来认识一下 GNOME 桌面上包含的内容。在这个桌面上包含以下几部分。

（1）桌面图标。是应用程序建立在桌面上的图形标识，使用桌面图标可以直接启动应用程序。

（2）应用程序组。包括应用程序、位置和系统 3 个菜单组。

①“应用程序”菜单组将系统中的应用程序进行分类组织，这些应用程序是安装系统时由用户自定义安装的，如图 3.7 所示。

②“位置”菜单组中包含访问系统中常见目录和设备的快捷方式，如访问用户主目录和光驱等。在图形界面中，使用位置菜单组通过鼠标操作即可访问这些常用的目录或设备，如图 3.8 所示。

③“系统”菜单组包含了一系列的系统管理与控制工具。包括首选项控制工具、系统管理工具、联机帮助以及注销与关机等选项，如图 3.9 所示。

（3）快捷方式。是一些应用程序的快捷方式链接，通过单击这些链接即可以启动应用程序，如图 3.10 所示。这里的快捷方式同桌面图标的区别在于，快捷方式通过单击即可启动，而桌面图标默认是通过双击启动的。而且，由于快捷方式不会像桌面图标一样被正在执行的应用程序所遮挡，所以在任何操作环境下均可以直接启动具有快捷方式的应用程序。

（4）显示桌面快捷方式。在屏幕的左下角有一个“显示桌面”的图标，通过单击该图标可以将当前桌面上启动的所有应用程序最小化，并显示桌面，如图 3.11 所示。

（5）桌面任务栏。显示的是当前系统启动的应用程序，如图 3.12 所示。



图 3.7 应用程序菜单



图 3.8 位置菜单组



图 3.9 系统菜单组



图 3.10 快捷方式

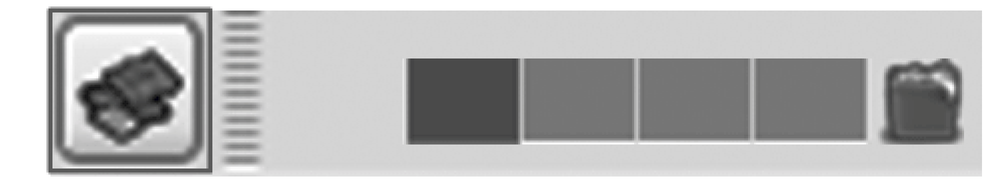


图 3.11 显示桌面



图 3.12 桌面任务栏

（6）工作面板与回收站。“工作面板”是指工作界面（桌面）。Linux 默认提供了 4 个工作桌面，当一个桌面被占用时，可以将应用程序在其他桌面打开，如图 3.13 所示。

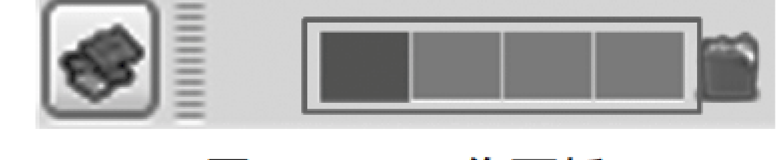


图 3.13 工作面板

“回收站”快捷方式可以进行打开回收站、清空回收站等操作。

（7）输入法、系统时钟与音频输出选项。用于控制输入法、显示与设置系统时钟、调整音频输出等操作。

在了解了 GNOME 的桌面构成后，希望大家试着进行一些桌面操作。具体的使用方法可自行参考其他资料，对于图形界面的使用不是本书的重点。

在 GNOME 图形界面中有一个应用程序需要向读者做重点介绍，这个应用程序就是“模拟终端”程序。

虽然系统管理工作主要是在命令行界面实现，但在 GNOME 界面中提供的模拟终端

gnome-terminal 也完全可以模拟出一个命令行界面，这个界面的显示更为清楚而且使用灵活，在虚拟终端显示不清楚的场合也常使用 gnome-terminal 作为终端界面来进行操作。

1. 启动 gnome-terminal

启动 gnome-terminal 可以在 GNOME 环境中使用鼠标右击桌面空白处，在弹出的快捷菜单中选择“打开终端”命令，如图 3.14 所示。也可以通过“应用程序”→“附件”→“终端”的方式打开终端。

2. gnome-terminal 终端界面

终端界面如图 3.15 所示。

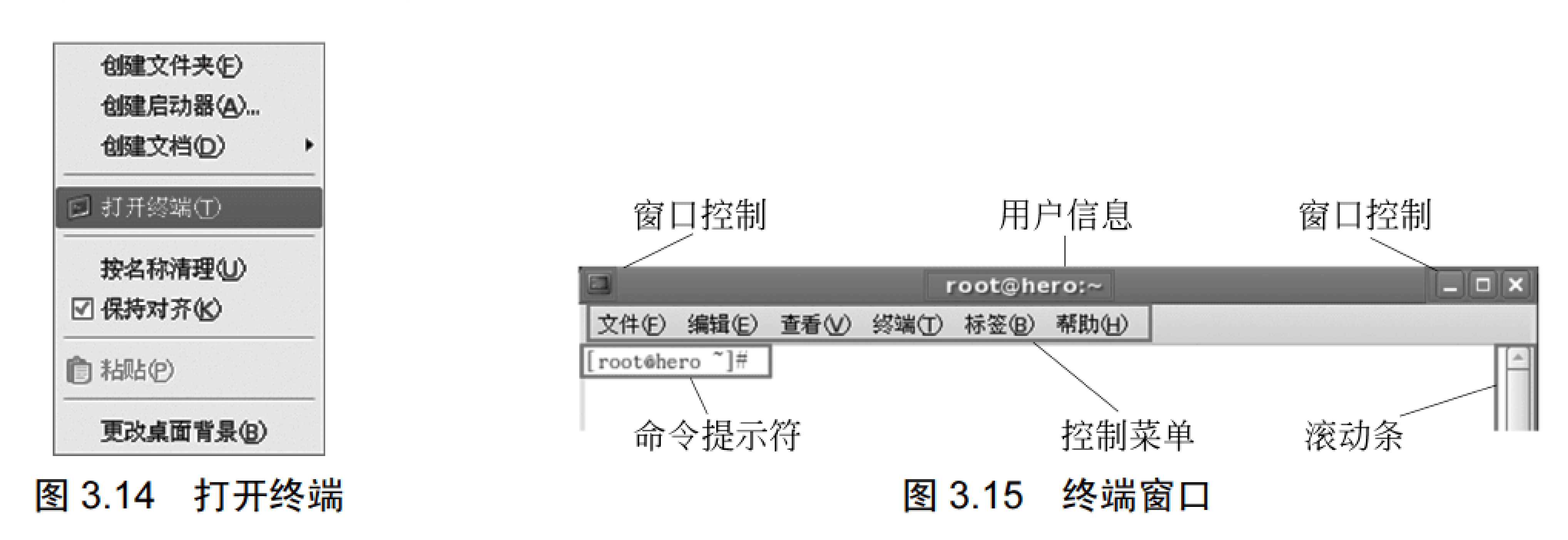


图 3.14 打开终端　　图 3.15 终端窗口

有过 Windows 操作系统使用经验的用户对这个窗口结构应该是很熟悉了。这里窗口控制用于控制窗口的最大化、最小化和关闭等操作。控制菜单是一系列的功能选择区域。用户信息中的“root@hero:~”表示当前是 root 这个用户，在 hero 这台计算机上，~表示是在 root 用户的用户主目录下。在命令提示符后输入命令即可执行命令。

3. gnome-terminal 终端控制

在 gnome-terminal 终端控制中主要介绍终端界面的标签操作，这一知识对于灵活使用这个终端界面很有帮助。

在 gnome-terminal 终端界面使用 Ctrl+Shift+T 组合键可以打开一个新的标签，使用 Ctrl+Shift+W 可以关闭当前标签，使用 Ctrl++（Ctrl 键和加号键）可以放大标签中的字体，使用 Ctrl+–（Ctrl 键和减号键）可以缩小标签中的字体。使用 Ctrl+Shift+N 可以重新打开一个终端界面，使用 Ctrl+Shift+Q 可以关闭终端界面。

3.1.2 KDE 环境下的终端程序

KDE 环境下的图形界面使用方法可自行参考其他资料，下面介绍 KDE 环境下的终端环境模拟程序。

在当前的 GNOME 环境下，选择“系统”应用程序组中的“注销”选项注销当前用户登录。在登录界面中将系统的会话定义为 KDE 之后，使用 root 用户重新登录系统。

在 KDE 环境下也存在一个终端模拟程序，称为 Konsole。在 KDE 环境下右击桌面

空白处，选择 Konsole 将打开该模拟终端环境。

3.1.3 图形界面下的注销与关机

1. 注销用户登录

在 GNOME 环境下注销用户登录的方法是选择“系统”菜单中的“注销”选项，这时系统会询问“立即注销系统吗？”，单击“注销”将实现系统的注销。

在 KDE 环境下注销用户登录的方法是选择 K 菜单中的“注销”，在弹出的操作方法菜单中选择“结束当前会话”，即实现注销当前用户登录的操作。

2. 关机操作

在 GNOME 环境下关闭计算机的方法是选择“系统”菜单中的“关机”选项，这时系统会询问“立即关闭系统吗？”，选择“关机”将关闭计算机。

在 KDE 环境下关机计算机的方法是选择 K 菜单中的“注销”，在弹出的操作方法菜单中选择“关闭计算机”，即实现关机的操作。

以上简单地介绍了图形界面的操作。尽管 Linux 的图形界面也很好，但这个界面对管理员的日常管理工作而言确实没有多大的帮助。

在了解完图形界面的基本操作后，下面详细介绍命令行界面的终端环境，这才是作为一个系统管理员所要真正了解的环境。

3.2 登录与虚拟终端

在 3.1 节中已经使用了 X Window 的图形环境，并选择图形界面登录操作系统，启动 Linux 后会看到 X Window 的登录画面。若无法顺利启动 X Window，则会看到文字模式（或称为主控模式，Console Mode）的登录画面。

建议对于 Linux 操作系统的学习要从文字模式入手，这有助于读者更快地理解和掌握 Linux 操作系统的特点，并且文字模式更贴近实际的工作环境。

如果当前系统停留在图形登录界面，使用 Ctrl+Alt+F1 这个组合键可以切换到文字模式的登录界面。

3.2.1 登录界面

文字模式的登录界面内容如下：

```
CentOS release 5.5  ( Final )
Kernel  2.6.18-194.el5  on  an i686

hero login : root
Password :
```

提示信息中各部分内容的意义如下。

CentOS release 5.5 （Final）：当前使用的操作系统的发行版本为 CentOS release 5.5，代号为 Final。

Kernel 2.6.18-194.el5：内核的版本是 2.6.18-194.el5。

on an i686：执行的计算机为使用相当于奔腾III以上等级 CPU 的 PC。

hero：这是这台计算机的主机名，该名称是在安装操作系统时由用户来定义的。如果用户在安装操作系统时没有定义主机名，则此处将显示默认主机名“localhost”。

login：是登录提示符，要求用户输入登录的账号名称。

Password：是密码提示符，要求用户输入登录账号的密码。

3.2.2 登录

输入账号名称为 root，系统紧接着以“Password：”询问账号的密码，在提示符后输入密码。在输入密码时，由于 Linux 的安全性设计，在屏幕上看不到输入的字符。root 账号及密码是在安装 Linux 操作系统的过程中已经设置好的。root 是超级用户，也就是系统管理员账号，以 root 登录后可以对系统做任意的修改，所以也是一个可能造成系统毁损的账号，使用时需要特别小心。Linux 操作系统不建议使用 root 账号登录系统进行日常的管理操作，而是建议使用普通账号登录进行系统管理，当需要使用 root 权限时再进行用户或权限的转换。但考虑到学习过程中的直观性和条理性，在学习过程中还是以 root 账号登录系统。

登录界面如下：

```
CentOS release 5.5  ( Final )
Kernel  2.6.18-194.el5  on  an i686

hero login : root
Password :
Last login: Sat Dec 18 03:48:29 on : tty1
[ root@hero ~ ]#
```

成功登录系统后看到的信息含义如下。

```
Last login: Sat Dec 18 03:48:29 on : tty1
```

若曾经以相同的账号登录过系统，这一行显示上一次登录的时间及登录的位置。在这里看到的 on tty1 是指 1 号虚拟终端。

```
[ root @ hero001 ~ ]#
```

这是系统 Shell 的提示符。“root @ hero001”表示用户 root 在 hero001 这台主机上；目前所在的目录为~（/root）；#为 root 账号的提示符，如果是以一般用户登录则提示符为$。提示符后就是用户输入命令的地方了。

在登录系统后也可以再执行 login 命令，以另一个账号登录。login 命令用于注销当前用户，并允许随时更换用户名登录系统。执行 login 命令后的登录界面信息如下：

```
[root@hero ~]# login

Cent OS release 5.5  ( Final )
Kernel  2.6.18-194.el5  on  an i686

hero login :
Password :
```

3.2.3 虚拟终端

Linux 是一个多任务、多用户的系统，即使是只有一台 PC，一样可以让多个用户同时在主机上执行工作。那么，如何让多个用户同时使用主机资源呢？Linux 采用的是虚拟终端机制。

之所以称为虚拟终端，是因为这些终端均是利用 PC 当前的键盘和显示器模拟出来的。在一个键盘上通过功能键的选择可以虚拟出多个终端来。在 Linux 系统内默认共有 6 个虚拟终端，虚拟终端也是一个终端，即同时可以有 6 个用户通过终端以文字模式登录 Linux 主机，使用系统资源。虚拟终端结构如图 3.16 所示。

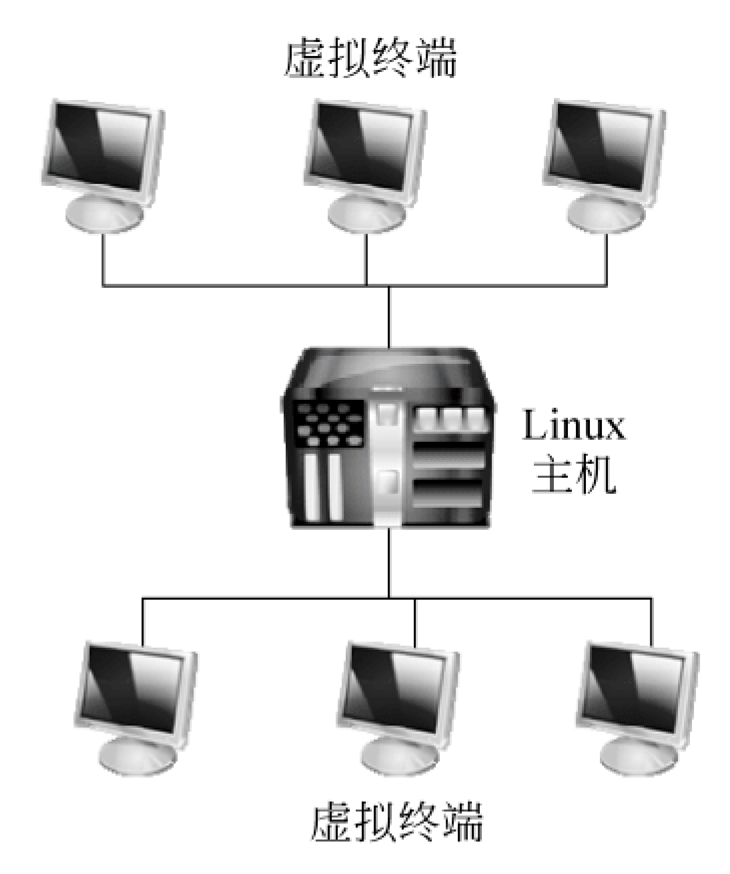


图 3.16 虚拟终端结构

虚拟终端在系统中分别以 tty1~tty5 来表示。可以使用 Alt+F1~Alt+F6 在虚拟终端间切换，使用 Alt+F7 将切换至 X Window 的图形终端界面，如图 3.17 所示。

当用户处于 X Window 图形终端环境下，需要切换到 tty1~tty6 中的任何一个文字模式的虚拟终端下时，可以使用 Ctrl+Alt+F1～Ctrl+Alt+F6 键切换。



图 3.17 虚拟终端间的切换

注意，Linux 使用 tty1~tty6 来表示虚拟终端，使用 tty0 来表示当前终端。

3.3 注销系统和关机

本节介绍如何注销系统，以保护数据及系统的安全，以及如何正确地关闭计算机。

3.3.1 注销系统

注销是指用户退出当前登录。由于 Linux 是一个多用户的操作系统，当一名用户使用完毕，注销系统可以避免其他用户访问自己的数据。

在 Linux 操作系统中可以使用 logout 命令来注销系统，注销后会回到登录界面。exit 命令和组合键 Ctrl+d 也可以实现注销系统的操作，但 exit 命令的含义是退出当前 Shell。对于什么是当前 Shell 这个问题会在后续章节讨论。

1. 用 logout 命令退出登录

```
[root@hero ~]# logout

CentOS release 5.5  ( Final )
Kernel  2.6.18-194.el5  on  an i686

hero login :
Password :
```

2. 用 exit 命令退出当前 Shell

```
[root@hero ~]# exit

CentOS release 5.5  ( Final )
Kernel  2.6.18-194.el5  on  an i686

hero login :
Password :
```

3.3.2 关机

由于 Linux 是一个多任务操作系统，即使没有用户在使用，系统也可能正在执行某些工作。Linux 会将一些工作中的文件以及系统的数据存放在内存的缓存区中，以提高系统的性能和管理能力，毕竟内存的读写速度要远远高于硬盘的读写速度。如果直接关闭计算机的电源会导致内存缓存区中的数据没有及时写入硬盘而丢失，进而破坏文件系统，所以必须要以命令执行正确的关机方式。标准的关机命令是 shutdown。

关闭系统是管理员 root 的工作，系统默认只有 root 有权力关闭系统，所以如果要关机，应先以 root 账号登录系统，然后使用 shutdown 命令进行关机。

下面举例说明 shutdown 命令的使用方法。

【示例】立即关闭计算机

```
# shutdown  -h  now
```

shutdown 是关机命令，-h 表示关闭计算机后直接关闭电源，now 表示立刻执行关机操作。注意，命令中的各项之间应使用空格分开。

【示例】5 分钟后重新启动计算机，并通知所有在线的用户。

```
# shutdown  -r  +5  "We'll poweroff in five minutes. "
```

shutdown 是关机命令，-r 表示关闭后重新启动计算机。+5 表示 5 分钟以后执行关机操作。“We’ll poweroff in five minutes.”是发送给所有在线用户的消息。

通过以上两个示例可见，shutdown 命令是一个很灵活的命令。当该命令的选项（-r、-h 等）不同时，该命令表出来的作用也不尽相同，并且该命令的时间表示方法也很灵活。下面介绍 shutdown 命令的语法结构。注意，shutdown 是我们接触的第一个带有选项的 Linux 命令，该命令的使用方法很具有代表性，通过对该命令的介绍能使我们逐步认识 Linux 的命令结构。

shutdown 命令格式如下：

shutdown [-选项] 时间 [提示信息]

其中，选项为：

-h：表示完成关机操作后切断电源。

-r：表示关闭系统后重新启动计算机。

-f：表示快速重新启动系统，重新启动系统时不会调用 fsck 检测文件系统。这样可以减少启动时间，提高启动速度。

-F：表示重新启动系统时强制使用 fsck 进行文件系统检测。

-c：表示取消关机程序。

-k：表示仅发送提示信息，不执行关机程序。

shutdown 命令中的时间表示方法见表 3.1。

表 3.1 shutdown 命令中的时间表示方法

格　　式	说　　明
hh:mm	以绝对时间表示，hh 为小时，mm 为分钟，如 08:30
+m	m 分钟后执行，如+5 表示 5 分钟后执行
now	now 表示+0，即立刻执行

shutdown 命令中的提示信息是通知在线用户系统将关闭的消息，该提示信息将被发送到所有在线终端。但是对于 shutdown 命令而言，提示信息却不是必要的参数，既可以添加提示信息，也可以不添加提示信息，这不会影响命令的执行。

下面再举几个 shutdown 命令的示例。

【示例】于凌晨 1:30 分重新启动计算机。

```
# shutdown -r 1:30
```

【示例】立刻快速重新启动计算机。

```
# shutdown -f now
```

【示例】15 分钟后关闭计算机，并通知用户。

```
# shutdown -h +15 "We'll poweroff in a quarter."
```

【示例】想取消刚刚的关机任务，并通知用户“关机程序已取消”。

```
# shutdown -c now "Shutdown program has cancelled."
```

3.4 Linux 系统基础

对于第一次接触 Linux 操作系统的用户而言，这是一个全新的操作系统，可能会完全对之感到陌生。所以本节将对 Linux 操作系统作一个简单的介绍。本节中涉及的很多问题都会在后续章节中详细论述，本节的目的仅仅是帮助读者初步认识 Linux。

3.4.1 文件目录与路径

计算机中的数据处理与数据存储都是以硬盘中的文件为基础的。在 Linux 系统中，硬盘中存储的数据是以文件和目录的方式整理保存的。下面介绍 Linux 的目录结构。

1. 单根文件系统

整个 Linux 系统内所有的文件与目录都存在于根目录（/）下，在 Linux 系统中根目录用“/”表示。根目录是文件系统的起点，且整个 Linux 系统只有一个根目录，所以称之为单根文件系统。Linux 系统中的其他分区及设备等均以文件的形式存储于根目录下。这一点与 Windows 操作系统不同，Windows 操作系统具有很多根目录，它的每一个分区都有一个根目录。Windows 和 Linux 的文件系统的比较如图 3.18 所示。

Windows文件系统 IDE1-MASTER		Linux文件系统 /dev/hda	
C:\		/dev/hda1	
D:\		/dev/hda2	
E:\		/dev/hda3	
扩展分区	F:\	扩展分区	/dev/hda5
	G:\		/dev/hda6
	H:\		/dev/hda7
	I:\		/dev/hda8
Windows操作系统中，每个分区均存在一个根目录，自成一个体系		Linux操作系统中，所有的设备均以文件形式存在于根目录“/”下	

图 3.18 Windows 和 Linux 文件系统的比较

对于单根文件系统而言，对于设备的访问与控制也是通过文件来实现的。这一点与在 Windows 中的概念不同，需要注意。

单根文件系统以根目录为起点，其他所有目录按层级关系排列在根目录下，形成一个类似于倒置的树的结构，称为目录树结构，如图 3.19 所示。

2. 工作目录与用户主目录

在目录树结构中，每个用户登录系统后会在哪个目录中呢？用户自己的文件保存在哪个目录中呢？这两个问题的答案是用户主目录。

用户主目录是用户登录系统后默认进入的目录，同时也是用户文件默认存储的目录。用户对自己的用户主目录具有完全控制的权限。普通用户的用户主目录默认位于/home 目录下，并以用户名作为目录名。如 user1 用户的用户主目录为/home/user1 目录。而管

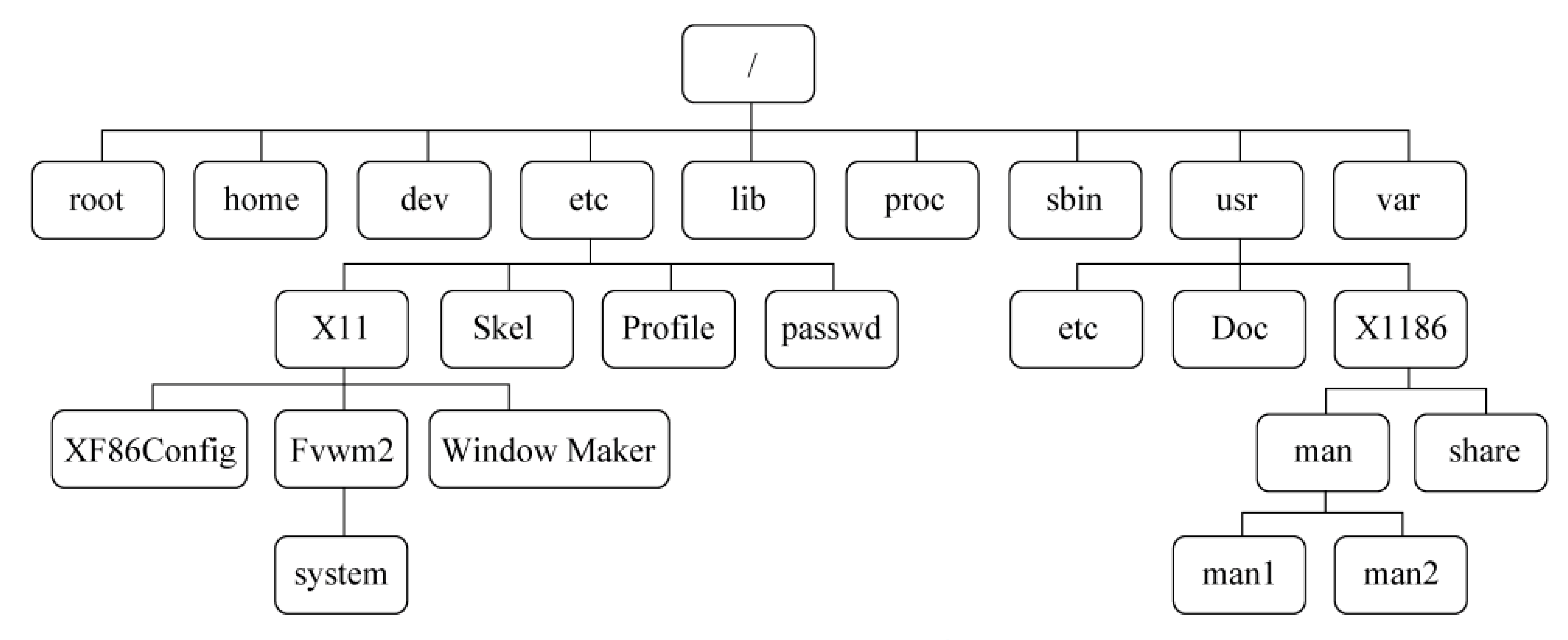


图 3.19　目录树结构

理员 root 的用户主目录是根目录下的 root 目录，即/root。

Linux 系统中常用~符号表示用户主目录，如 $cd ~ 这个命令表示进入当前用户的用户主目录。

工作目录是指用户当前所处的目录。这个目录可以是用户主目录，也可以是其他目录。用户常常在不同的目录间进行操作，这时用户的工作目录也会频繁地发生变化。

3．路径

在 Linux 的文件目录系统里以路径来指向一个文件或目录，例如，/etc/passwd 就是一个路径，它是指“/目录下的 etc 目录下的 passwd 文件”。在/etc/passwd 中第一个/表示根目录，其后的/是目录分隔符（注意这一点，在以后所有的路径表示中第一个/表示根目录，其后的所有/都是目录分隔符）。

在了解了什么是路径后，需要明确什么是相对路径，什么是绝对路径。

绝对路径是以“/”根目录为起点定位目标文件或目录位置的路径。如/etc/passwd 就是绝对路径。绝对路径具有引用准确、含义清晰的特点。

相对路径是以当前工作目录为起点定位目标文件或目录位置的路径。

以图 3.20 为例，若用户当前的工作目录为/usr/X1186，这时要访问 man1 文件，如果使用相对路径，则可以表示为“man/man1”这样一个路径。

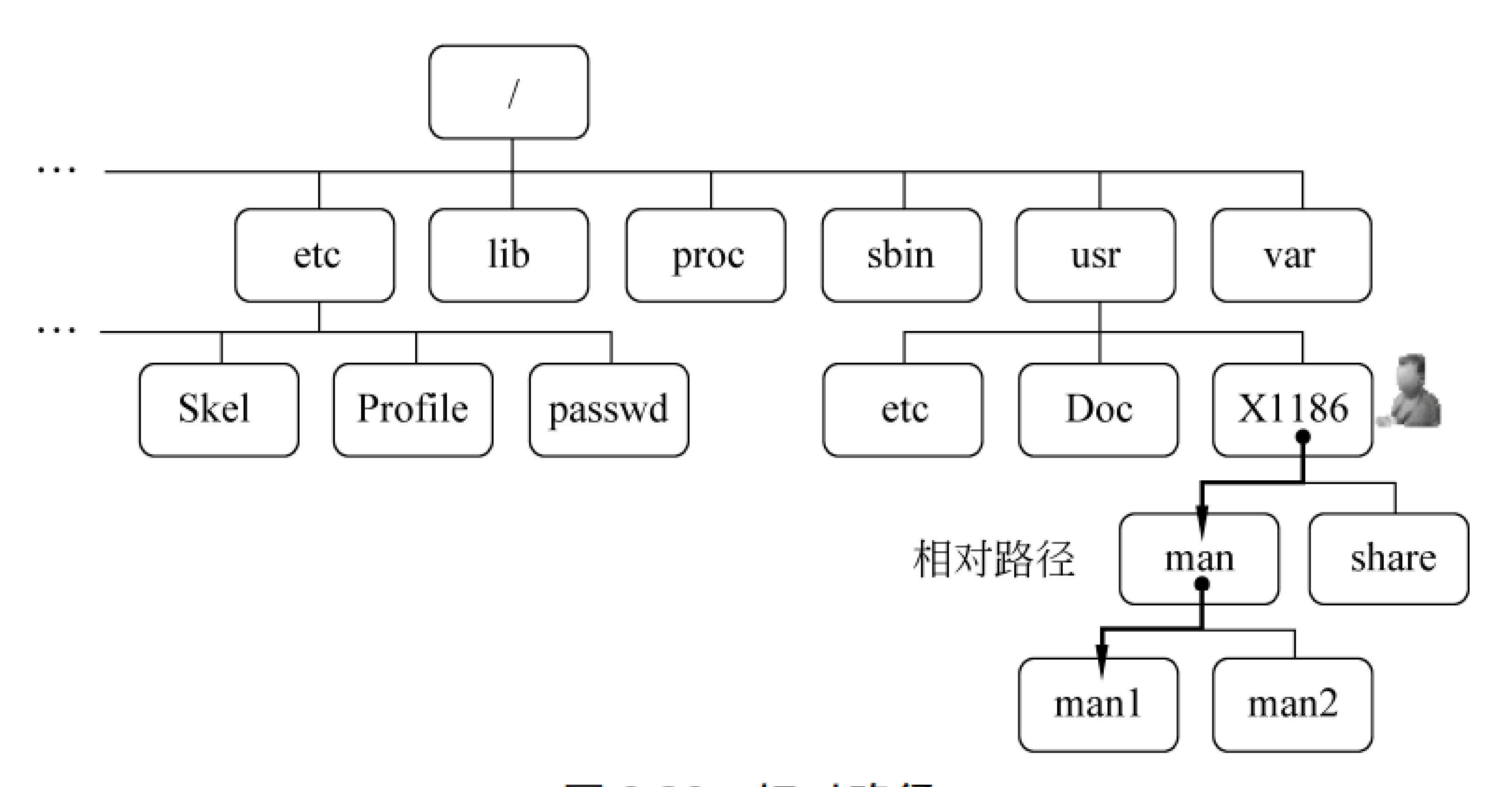


图 3.20　相对路径

相对路径在使用时比绝对路径简短，但相对路径的应用受当前工作目录的限制。如果当前的工作目录在/etc 目录下，则访问 man1 文件还是使用绝对路径更为方便。

在相对路径的表示方法中，有两个目录值得注意，就是“.”和“..”。“.”表示当前目录，“..”表示上级目录。所以“../myfile”这个路径表示的是上级目录中的 myfile 文件。

3.4.2 用户与操作系统之间的界面——Shell

对于一个操作系统而言，一般是由内核、环境、文件系统和网络接口 4 个部分组成的。其中内核是操作系统的核心与灵魂，整个操作系统是在内核的控制下完成工作的。那么，用户是如何使用操作系统内核的呢？由于内核过于重要，且可交互性不好，所以用户是无法直接使用内核的。当用户需要执行相关操作时是通过用户环境向内核提交请求的，内核执行完相关操作后会将结果通过用户环境反馈给用户。也就是说在用户与操作系统内核之间还存在着一个环境层，如图 3.21 所示。

用户

环境

内核

图 3.21 内核、环境与用户

那么操作系统的环境是什么呢？对于 GUI（图形用户界面）而言就是大家所使用的 GNOME 或 KDE；而对于文字模式而言，这个环境就是 Shell。Shell 是一个应用程序，是文字模式下用户与内核之间的界面，负责将用户命令解释为内核编码，并交由内核执行。同时，也会将内核的执行结果解释为用户能读懂的结果反馈给用户。可见，这里 Shell 主要起到了一种命令解释器的作用。

Shell 除了具有命令解释器的作用外，还具有一定的程序设计能力。利用 Shell 所提供的控制结构可以很灵活地利用 Shell 脚本完成日常的系统管理工作。

在 Linux/UNIX 操作系统下有多种 Shell，CentOS 默认的 Shell 为 bash。有关于 Shell 的详细介绍将在后续章节展开，这里只要求理解什么是 Shell 即可。

3.5 基 本 操 作

本节介绍 Linux 下的基本操作方法，包括常用的快捷键以及几个基本指令的使用，以使读者能很快地熟悉 Linux 的操作。

3.5.1 常用的快捷键

在终端的文字模式下操作 Linux 会经常用到快捷键，例如，按下 Ctrl+C 键可以中断程序。这些快捷键的设置是可以改变的，但通常用户比较熟悉的默认用法包括以下几个。

Enter：输入完成。使用 Enter 键就会把输入的文字传递到 Shell 进行分析和解释。

Ctrl+C：中断正在执行的程序。

Ctrl+Z：挂起正在执行的程序。利用该快捷键可以将正在运行的程序挂起，当需要继续运行被挂起的程序时，可以使用工作调度命令来继续运行。这与 Ctrl+C 不同，Ctrl+C

是中断正在运行的程序，而不可能再要求被中断的程序继续运行了。

Ctrl+D：退出当前的 Shell，该快捷键的功能相当于 exit。

3.5.2 基本操作指令

下面是一些 Linux 下的基本操作以及一些常用命令的用法。

1. 命令的联机帮助

在 Linux 操作系统中有很多命令，每个命令又有很多的选项与参数。应该如何掌握这么多的命令、选项和参数呢？Linux 提供了联机帮助，详细说明了每个命令、选项与参数的使用方法。熟练地使用联机帮助对我们了解和使用 Linux 操作系统是大有裨益的。

Linux 中的联机帮助命令是 man。man 是 manual 的简写，是应用程序开发人员提供给用户的帮助文件。

1）man 的使用方法

```
man [选项] {命令名|字符串}
```

要查看 shutdown 命令的使用方法，命令如下：

```
[root@hero ~]# man shutdown
```

执行该命令后系统显示的信息如下：

```
SHUTDOWN(8)   Linux System Administrator's Manual    SHUTDOWN(8)
NAME
       shutdown - bring the system down
SYNOPSIS
       /sbin/shutdown [-t sec] [-arkhncfFHP] time [warning-message]
DESCRIPTION
       shutdown  brings  the system down in a secure way.
       All logged-in users are notified that the system is going down,
       and  login(1)is  blocked.
       It is possible to shut the system down immediately or after a specified
       …
OPTIONS
       -a    Use /etc/shutdown.allow.
       -t sec Tell  init(8)  to wait sec seconds between sending
       processes the warning and the kill signal, before  changing  to
       another run-level.
         …
ACCESS CONTROL
       shutdown  can  be  called from init(8) when the magic keys
       CTRL-ALT-DEL are pressed, by creating an appropriate  entry  in
```

```
      /etc/inittab. This means that everyone who has physical access to
      the console keyboard can
        …
FILES
      /fastboot
      /etc/inittab
        …
NOTES
      A lot of users forget to give the time argument and are then
      puzzled by the error message shutdown produces. The time argument
      is mandatory; in 90 percent of all cases this argument will be the
      word now.
        …
AUTHOR
      Miquel van Smoorenburg, miquels@cistron.nl
SEE ALSO
      fsck(8), init(8), halt(8), poweroff(8), reboot(8)
                      November 12, 2003                SHUTDOWN(8)
```

显示的说明内容非常详细。在屏幕的最下端看到光标在“:”后，表示显示的内容超过了这一页的范围，这时按空格键就可以看到下一页的内容。使用 Page Down 和 Page Up 键可以上下翻页查看说明。当查看完成时，按 q 键退出 man 程序。

2）man 手册页的结构

man 手册页由关键字分隔为多个部分，下面介绍一下手册页的结构。

命令（章节号）：在联机手册的第一页的第一行。

```
SHUTDOWN(8)   Linux System Administrator's Manual    SHUTDOWN(8)
```

由上可见 shutdown 命令所在的章号是 8，也就是管理员命令。为什么 8 就是管理员命令呢？这个问题在有关 man 的章节中再做介绍。这里先知道第一行所传达的信息即可。

NAME：命令或数据文件的功能简介。

```
NAME
      shutdown - bring the system down
```

SYNOPSIS：命令语法简介。

```
SYNOPSIS
      /sbin/shutdown [-t sec] [-arkhncfFHP] time [warning-message]
```

DISCRIPTION：命令的详细说明（应该仔细参考）。

```
DESCRIPTION
      shutdown  brings  the system down in a secure way.  All logged-in
      users are notified that the system is going down, and  login(1)
      is blocked.
```

```
    It is possible to shut the system down immediately or after a specified
     …
```

OPTIONS：针对 SYNOPSIS 部分，举例说明所有可用的参数。

```
OPTIONS
      -a      Use /etc/shutdown.allow.
      -t sec Tell  init(8)  to wait sec seconds between sending processes
             the warning and the kill signal, before  changing  to  another
             run-level.
       …
```

FILES：与该程序相关的文件，包括程序使用的、参考的或链接的某些文件。

```
FILES
      /fastboot
      /etc/inittab
       …
```

SEE ALSO：这个命令可以参考的其他文件。

```
SEE ALSO
      fsck(8), init(8), halt(8), poweroff(8), reboot(8)
```

AUTHOR：命令或应用程序的作者及联系方式。

```
AUTHOR
      Miquel van Smoorenburg, miquels@cistron.nl
```

除了上述介绍的关键字之外，不同的命令或配置文件还有不同的关键字。可以使用 man 多查找几个命令的帮助，以熟悉 man 页面的结构。

【示例】查看 date 命令的联机帮助手册。

```
# man  date
```

【示例】查看 cal 命令的联机帮助手册。

```
# man  cal
```

【示例】查看/etc/passwd 文件的联机帮助手册。

```
# man  5  passwd
```

在命令 man 5 passwd 中，5 的作用是什么呢？shutdown 命令的章号为什么是 8 呢？这就涉及 man 的章节结构问题了。对于操作系统而言，应用程序帮助的种类很多，为了能更好地分类管理帮助信息，man 将应用程序的帮助文件分为 8 类，称为 8 个章，见表 3.2。

表 3.2　帮助文件的章节结构

章号	包 含 内 容	示　　例
1	普通用户可使用的命令或可执行文件的帮助信息	man　date
2	系统内核可调用的函数与工具的帮助信息	man　idle
3	常用的函数（function）与函数库（library）的信息	man　ynl
4	设备文件的说明信息	man　zero
5	配置文件说明	man　yp.conf
6	游戏的帮助信息	man　6　intro
7	惯例与协议等内容的帮助信息	man　x25
8	系统管理员命令的帮助信息	Man　shutdown

3）man page 的控制方式

在 man 帮助页面中，如何实现页面的翻动？如何实现对字符串的查找等操作呢？下面介绍页面的相关控制方法。

空格键：向下翻一页。

Page Down：向下翻一页。

Page Up：向上翻一页。

Enter：向下滚动一行。

Home：到第一页。

End：到最后一页。

/字符串：向下搜索字符串。如需要在 shutdown 的 man 帮助中查找-h 选项的说明，这时只需要输入“/-h”，man 会自动在页面中查找“-h”字符串并以反白的形式显示出来。但是要注意，这是向下搜索字符串，即由当前光标的位置向下搜索。

？字符串：向上搜索 string 字符串。其功能也是搜索字符串，但与向下搜索字符串不同的是，“？字符串”是向上搜索字符串，即由当前光标的位置向上搜索。

n：继续上一个字符串查找。

N：继续上一个查找（反向查找）。如果上一个搜索是向下查找，则 N 实现的是向上查找。

q：退出 man 帮助。

man 为系统中众多的命令和配置文件提供联机帮助。当系统中存在命令与配置文件同名的情况时，该如何确定究竟是使用命令的帮助还是配置文件的帮助呢？如 passwd 既是一个命令名称，也是一个配置文件名称。如果使用“# man passwd”这个命令，则默认显示的是 passwd 的命令帮助信息。那么如何获得 passwd 配置文件的帮助信息呢？从前面的实例我们知道还需要添加章节号，即使用#man　5　passwd。因为第 5 章是对配置文件的帮助信息，所以 man 会将 passwd 配置文件的帮助信息显示出来。

4）man 命令的完整说明

```
man  [选项]  {命令名 | 字符串}
```

选项说明：
章号：按照特定章号显示命令的帮助信息。
-f：显示命令在 man 帮助中的所有分布情况。
-k：显示包含特定字符串的 man 帮助信息。
【示例】查看 tty 设备的帮助信息。

```
# man 4 tty
```

【示例】查看 shutdown 在 man 中的分布情况，即除作为命令存在外还有哪些应用。

```
[root@hero ~]# man -f shutdown
shutdown             (2)  - shut down part of a full-duplex connection
shutdown            (3p)  - shut down socket send and receive operations
shutdown             (8)  - bring the system down
```

由输出可见，shutdown 除了作为管理员命令（帮助手册第 8 章）存在之外，在系统内核函数中也包括一个 shutdown 函数（帮助手册第 2 章），另外，在帮助手册第 3 章常用函数中也包含一个 shutdown 函数。因此，-f 选项的作用是显示一个命令在 man 帮助信息中的所有分布情况。

【示例】查看包含 ksh 字符串的帮助信息有哪些。

```
[root@hero ~]# man -k ksh
.if nZ=0 { sh [ksh93] (1)  - shell, the standard/restricted command and
programming language
.if nZ=1 { ksh [ksh93] (1)  - KornShell, a standard/restricted command and
programming language
.if nZ=2 { ksh93 [ksh93] (1) - KornShell, a standard/restricted command
and programming language
ksh             (rpm) - 原始 ATT Korn Shell
pammasksharpen  (1)  - Sharpen an image via an unsharp mask
pfksh93 [ksh93] (1) - KornShell, a standard/restricted command and
programming language
pfksh [ksh93]   (1) - KornShell, a standard/restricted command and
programming language
```

注意这个输出结果，显示了包含 ksh 字符串的相关命令和配置文件的帮助情况。-k 选项的作用是提供了一种模糊的查找方式，即把包含特定字符串的帮助信息全部显示出来。这一点对于日常管理工作而言是很方便的。如果忘了命令的拼写，而只记得是以一个或几个字符开头的，则使用-k 选项会将包含这个开头字符串的所有命令的帮助显示出来，以方便查找。

2. 显示和设置日期与时间命令

显示和设置系统日期和时间的命令是 date。可以利用 man date 命令查看 date 命令的使用方法。

```
[root@hero ~]# man date
DATE(1)        User Commands          DATE(1)

NAME
      date - print or set the system date and time

SYNOPSIS
      date [OPTION]... [+FORMAT]
      date [-u|--utc|--universal] [MMDDhhmm[[CC]YY][.ss]]

DESCRIPTION
      Display the current time in the given FORMAT, or set the system date.

      -d, --date=STRING
            display time described by STRING, not 'now'

      -f, --file=DATEFILE
...
```

通过输出可见，date 是帮助手册第 1 章的命令，即普通用户命令，用于“print or set the system date and time”，即打印或设置系统日期和时间。

1）利用 date 命令显示系统日期和时间

【语法】**date　[+格式]**

【示例】显示当前系统的日期和时间。

date

```
[root@hero ~]# date
Mon Mar 28 14:33:16 CST 2011
```

date 命令如果不添加任何参数，将直接显示系统的日期和时间信息，其输出格式为“星期　月份　日期　时间　时区　年份”。从示例中可见，当前的系统时间为 2011 年 3 月 28 日，星期一，14:33:16，系统时区为 CST （与 UTC 偏移量为 06:00 的中部标准时间）。

【示例】以特定的格式显示系统日期和时间。

命令的格式为：

date　+格式控制字符

其中的格式控制字符如下：

%Y　表示年份；

%m　表示月份；

%d　表示日期；

%H　表示小时；

%M　表示分钟。

例如，以年/月/日的格式显示系统日期：

```
[root@hero ~]# date +%Y/%m/%d
2011/03/28
```

以“小时:分钟”格式显示系统时间的命令如下：

```
[root@hero ~]# date +%H:%M
14:35
```

以“年-月-日 小时:分钟”格式显示系统的日期和时间的命令如下：

```
[root@hero ~]# date "+%Y-%m-%d %H:%M"
2011-03-28 14:38
```

date 命令可使用的时间格式控制符号有很多，常用的格式控制符号见表 3.3。

表 3.3　data 命令常用的时间格式控制字符

选　项	作　用	选　项	作　用
%Y	显示年份	%M	显示分钟
%m	显示月份	%S	显示秒钟
%d	显示日期	%p	显示 AM 与 PM 标识
%D	以“月/日/年”格式显示时间	%Z	显示时区
%A	显示星期全称	MM	表示月份
%a	显示星期简写	DD	表示日期
%B	显示月份全称	hh	表示小时
%b	显示月份简写	mm	表示分钟
%G	显示 4 位年份值	CC	表示年份的前两位值
%g	显示两位年份值	YY	表示年份的后两位值
%H	显示小时（24 小时制）	.ss	表示秒数
%I	显示小时（12 小时制）		

2）利用 date 命令修改系统日期和时间

【语法】**date　MMDDhhmmYYYY**

其中：

MM 表示月份；

DD 表示日期；

hh 表示小时；

mm 表示分钟；

YYYY 表示年份。

【示例】显示和修改系统日期和时间。

```
[root@hero ~]# date
Mon Mar 28 14:42:35 CST 2011
[root@xinya ~]# date 032909462011
Tue Mar 29 09:46:00 CST 2011
```

有关 date 命令的其他格式控制字符可以根据 man 手册中的说明来灵活使用，这里就不一一介绍了。

3. 显示日历命令

显示日历的命令是 cal，该命令可用于显示各时期的日历信息。

【语法】**cal　[月份] [年份]**

【示例】显示本月份的日历。

```
[root@hero ~]# cal
       March 2011
Su  Mo  Tu  We  Th  Fr  Sa
         1   2   3   4   5
 6   7   8   9  10  11  12
13  14  15  16  17  18  19
20  21  22  23  24  25  26
27  28  29  30  31
```

cal 命令不添加任何参数时会显示系统当前月份的日历，也可以使用 cal 命令显示特定年份与特定月份的日历。

【示例】显示 2011 年全年的日历。

```
[root@hero ~]# cal 2011
```

这里使用的是“年份”参数。

【示例】显示 2011 年 10 月份的日历。

```
[root@hero ~]# cal 10 2011
```

这里使用的是“月份　年份”这样的参数。

4. 任意精度的计算器

Linux 的文字模式中提供了一个任意精度的计算器——bc，其使用方法是一种交互式操作，这与前面介绍的命令的使用方法稍有不同。bc 计算器支持加法+、减法–、乘法×、除法÷、取模运算%和指数运算^等计算。

要启动 bc 计算器，直接使用 bc 命令。

```
[root@hero ~]# bc
bc 1.06
```

```
Copyright 1991-1994, 1997, 1998, 2000 Free Software Foundation, Inc.
This is free software with ABSOLUTELY NO WARRANTY.
For details type 'warranty'.
```

显示完版本信息后，bc 程序开始等待用户输入。这时可以输入要计算的数学表达式，如：

```
2^8  数学表达式，计算 2^8
256  计算结果

96/8  数学表达式，计算 96÷8
12    计算结果

35/4  数学表达式，计算 35÷4
8     计算结果（忽略了小数部分）
```

通过以上计算，我们发现。在默认的情况下 bc 计算器只进行整数运算，如果运算过程中出现小数，则 bc 计算器忽略小数位。当计算要求精度很高时就需要调整 bc 计算器的精度设置。使用 scale=n（n 为小数点后的位数）命令来设置 bc 计算器的精度，如：

```
scale=6  设置计算精度为小数点后 6 位。
35/4
8.750000
```

当完成计算后可以使用 quit 命令退出 bc 计算器。

```
quit
[root@hero ~]#
```

以上 3 个命令 date、cal 和 bc 分别代表了 Linux 中 3 种典型的命令的使用方法。接下来介绍几个在系统使用过程中常用的命令。

5. 显示目录内容

1）显示当前工作目录下内容

```
[root@hero /]# ls
bin  dev  home  lost+found  misc  net  proc  sbin     srv  tmp  var
boot etc  lib   media       mnt   opt  root  selinux  sys  usr
```

在 CentOS 中，默认的情况下会看到不同颜色的文件名称，一般目录使用蓝色表示，普通文件使用白色表示，可执行文件使用绿色表示。利用这些颜色可以方便地区分文件和目录。

2）以长格式显示目录和文件信息

```
[root@hero /]# ls -l
total 142
```

```
drwxr-xr-x   2  root root  4096 Mar 15  10:14   bin
drwxr-xr-x   4  root root  1024 Mar 14  22:33   boot
drwxr-xr-x  12  root root  3980 Mar 28  14:30   dev
drwxr-xr-x 102  root root 12288 Mar 28  14:31   etc
drwxr-xr-x  12  root root  4096 Mar 15  09:39   home
drwxr-xr-x  13  root root  4096 Mar 15  10:12   lib
drwx------   2  root root 16384 Mar 14  22:18   lost+found
drwxr-xr-x   3  root root  4096 Mar 28  14:31   media
drwxr-xr-x   2  root root     0 Mar 28  14:30   misc
drwxr-xr-x   2  root root  4096 Jan 27  2010    mnt
drwxr-xr-x   2  root root     0 Mar 28  14:30   net
drwxr-xr-x   2  root root  4096 Jan 27  2010    opt
dr-xr-xr-x 134  root root     0 Mar 28  14:29   proc
drwxr-x---  19  root root  4096 Mar 28  14:31   root
…
```

这里在 ls 命令之后使用了-l 这个选项，-l 的作用就是以长格式显示目录和文件信息。在一些 Linux 发行版中 ls -l 命令会被别名链接为 ll，也就是 ll 命令与“ls –l”命令的作用是相同的。

3）显示所有文件和目录信息，包括隐藏文件和目录

```
[root@hero ~]# ls -a
.                .config          .gstreamer-0.10     .tcshrc
.                .cshrc           .gtkrc-1.2-gnome2   .xsession-errors
.ICEauthority    .dmrc            .lesshst            Desktop
.Trash           .eggcups         .metacity           anaconda-ks.cfg
.bash_history    .gconf           .mozilla            install.log
.bash_logout     .gconfd          .nautilus           install.log.syslog
.bash_profile    .gnome           .recently-used.xbel mbox
.bashrc          .gnome2          .redhat             test
.chewing         .gnome2_private .scim
```

当直接使用 ls 命令时，当前工作目录下有 2 个目录和 4 个文件，当使用“ls –a”命令时发现当前工作目录下多出很多以“.”开头的文件和目录，这些以“.”开头的文件和目录是隐藏文件。

4）显示其他目录中的文件和子目录

ls 目录名

```
[root@hero ~]# ls /boot/grub
device.map       grub.conf           minix_stage1_5      stage2
e2fs_stage1_5    iso9660_stage1_5    reiserfs_stage1_5   ufs2_stage1_5
fat_stage1_5     jfs_stage1_5        splash.xpm.gz       vstafs_stage1_5
ffs_stage1_5     menu.lst            stage1              xfs_stage1_5
```

上述命令查看的是/boot/grub/目录中的文件和子目录。

5）以长格式的形式查看其他目录中的所有文件，包括隐藏文件

ls - a l 目录名

```
[root@hero ~]# ls -al /boot/grub/
total 261
drwxr-xr-x  2 root root    1024 Mar 14 22:53 .
drwxr-xr-x  4 root root    1024 Mar 14 22:33 ..
-rw-r--r--  1 root root      63 Mar 14 22:53  device.map
-rw-r--r--  1 root root    7584 Mar 14 22:53  e2fs_stage1_5
-rw-r--r--  1 root root    7456 Mar 14 22:53  fat_stage1_5
-rw-r--r--  1 root root    6720 Mar 14 22:53  ffs_stage1_5
```

注意，对于一个命令而言，不同的选项可以联合使用，以便灵活而高效地完成相关的操作。

6）以易读的方式显示文件的容量信息

文件和目录的容量信息默认是以字节为单位进行计数的，当文件容量很大时容量信息将变得不好统计，可以使用-h 选项，这时，文件或目录的容量信息将以最合适的单位显示出来，方便读取。

ls -h l   目录名

```
[root@hero ~]# ls -l /boot
total 6283                                          ← 容量单位为字节
-rw-r--r-- 1 root root   967675 Apr  3  2010 System.map-2.6.18-194.el5
-rw-r--r-- 1 root root    69593 Apr  3  2010 config-2.6.18-194.el5
drwxr-xr-x 2 root root     1024 Mar 14 22:53 grub
-rw------- 1 root root  3275197 Mar 14 22:33 initrd-2.6.18-194.el5.img
drwx------ 2 root root    12288 Mar 14 22:19 lost+found
-rw-r--r-- 1 root root    80032 Mar 13  2009 message
-rw-r--r-- 1 root root   110979 Apr  3  2010 symvers-2.6.18-194.el5.gz
-rw-r--r-- 1 root root  1875796 Apr  3  2010 vmlinuz-2.6.18-194.el5
[root@hero ~]# ls -lh /boot
total 6.2M                                          ←使用"-h"选项后,容量信息添加了单位
-rw-r--r-- 1 root root 945K    Apr  3  2010 System.map-2.6.18-194.el5
-rw-r--r-- 1 root root 68K     Apr  3  2010 config-2.6.18-194.el5
drwxr-xr-x 2 root root 1.0K    Mar 14 22:53 grub
-rw------- 1 root root 3.2M    Mar 14 22:33 initrd-2.6.18-194.el5.img
drwx------ 2 root root 12K     Mar 14 22:19 lost+found
-rw-r--r-- 1 root root 79K     Mar 13  2009 message
-rw-r--r-- 1 root root 109K    Apr  3  2010 symvers-2.6.18-194.el5.gz
-rw-r--r-- 1 root root 1.8M    Apr  3  2010 vmlinuz-2.6.18-194.el5
```

如果需要更改当前的工作目录，可以使用 cd 命令。

7）查看目录本身的信息

ls 默认是查看当前工作目录下的文件和子目录的信息，如果需要查看当前目录本身

的信息，需要使用“-d”选项。该选项用于显示目录本身的相关信息。

```
[root@hero ~]# ls -l /boot/grub
total 257
-rw-r--r-- 1 root root         63 Mar 14 22:53 device.map
-rw-r--r-- 1 root root       7584 Mar 14 22:53 e2fs_stage1_5
-rw-r--r-- 1 root root       7456 Mar 14 22:53 fat_stage1_5
-rw-r--r-- 1 root root       6720 Mar 14 22:53 ffs_stage1_5
-rw------- 1 root root        605 Mar 14 22:53 grub.conf
-rw-r--r-- 1 root root       6720 Mar 14 22:53 iso9660_stage1_5
-rw-r--r-- 1 root root       8192 Mar 14 22:53 jfs_stage1_5
lrwxrwxrwx 1 root root         11 Mar 14 22:53 menu.lst -> ./grub.conf
-rw-r--r-- 1 root root       6880 Mar 14 22:53 minix_stage1_5
-rw-r--r-- 1 root root       9248 Mar 14 22:53 reiserfs_stage1_5
-rw-r--r-- 1 root root      55808 Mar 13  2009 splash.xpm.gz
-rw-r--r-- 1 root root        512 Mar 14 22:53 stage1
-rw-r--r-- 1 root root     104988 Mar 14 22:53 stage2
-rw-r--r-- 1 root root       7072 Mar 14 22:53 ufs2_stage1_5
-rw-r--r-- 1 root root       6272 Mar 14 22:53 vstafs_stage1_5
-rw-r--r-- 1 root root       8904 Mar 14 22:53 xfs_stage1_5
[root@hero ~]# ls -ld /boot/grub
drwxr-xr-x 2 root root       1024 Mar 14 22:53 /boot/grub
```

注意，ls 默认显示的是目标目录下的文件和子目录信息，所以“ls -l /boot/grub”显示的是/boot/grub 目录下的文件和子目录信息。当使用-d 选项时，ls 显示的是目标目录/boot/grub 本身的信息。

8）排序输出目录内容

① 默认是按文件名由大到小输出，如：

```
[root@hero ~]# ls -l
total 76
drwxr-xr-x 3 root root  4096 Mar 16 08:55  Desktop
-rw------- 1 root root  1250 Mar 14 22:53  anaconda-ks.cfg
-rw-r--r-- 1 root root 29387 Mar 14 22:53  install.log
-rw-r--r-- 1 root root  4434 Mar 14 22:46  install.log.syslog
-rw------- 1 root root  9285 Mar 25 00:21  mbox
drwxr-xr-x 9 root root  4096 Mar 25 01:05  test
```

② 按文件名由小到大输出，使用-r 选项。-r 选项的作用是将排序结果反向输出。

```
[root@hero ~]# ls -lr
total 76
drwxr-xr-x 9 root root  4096 Mar 25 01:05     test
-rw------- 1 root root  9285 Mar 25 00:21     mbox
```

```
-rw-r--r-- 1 root root   4434 Mar 14 22:46     install.log.syslog
-rw-r--r-- 1 root root  29387 Mar 14 22:53     install.log
-rw------- 1 root root   1250 Mar 14 22:53     anaconda-ks.cfg
drwxr-xr-x 3 root root   4096 Mar 16 08:55     Desktop
```

③ 按时间顺序由新到旧排序输出，使用-t 选项。-t 选项的作用是按由新到旧的时间顺序排序输出，如：

```
[root@hero ~]# ls -lt
total 76
drwxr-xr-x 9 root root   4096  Mar 25 01:05    test
-rw------- 1 root root   9285  Mar 25 00:21    mbox
drwxr-xr-x 3 root root   4096  Mar 16 08:55    Desktop
-rw------- 1 root root   1250  Mar 14 22:53    anaconda-ks.cfg
-rw-r--r-- 1 root root  29387  Mar 14 22:53    install.log
-rw-r--r-- 1 root root   4434  Mar 14 22:46    install.log.syslog
```

④ 按时间顺序由旧到新排序输出，这是-t 选项的反向操作，可以将-t 选项与-r 选项结合使用，得到的就是-t 选项的反向输出结果。

```
[root@hero ~]# ls -ltr
total 76
-rw-r--r-- 1 root root   4434 Mar 14 22:46     install.log.syslog
-rw-r--r-- 1 root root  29387 Mar 14 22:53     install.log
-rw------- 1 root root   1250 Mar 14 22:53     anaconda-ks.cfg
drwxr-xr-x 3 root root   4096 Mar 16 08:55     Desktop
-rw------- 1 root root   9285 Mar 25 00:21     mbox
drwxr-xr-x 9 root root   4096 Mar 25 01:05     test
```

⑤ 按文件的容量由大到小排序输出，使用-S 选项。-S 选项的作用是按文件的容量由大到小排序输出，如：

```
[root@hero ~]# ls -hlS
total 76K
-rw-r--r-- 1 root root 29K   Mar 14 22:53  install.log
-rw------- 1 root root 9.1K  Mar 25 00:21  mbox
-rw-r--r-- 1 root root 4.4K  Mar 14 22:46  install.log.syslog
drwxr-xr-x 3 root root 4.0K  Mar 16 08:55  Desktop
drwxr-xr-x 9 root root 4.0K  Mar 25 01:05  test
-rw------- 1 root root 1.3K  Mar 14 22:53  anaconda-ks.cfg
```

⑥ 按文件的容量由小到大排序输出。这是-S 选项的反向操作，可以将-S 选项与-r 选项联合使用，就得到-S 选项的反向输出结果。

```
[root@hero ~]# ls -hlSr
total 76K
```

```
-rw------- 1 root root 1.3K  Mar 14 22:53  anaconda-ks.cfg
drwxr-xr-x 9 root root 4.0K  Mar 25 01:05  test
drwxr-xr-x 3 root root 4.0K  Mar 16 08:55  Desktop
-rw-r--r-- 1 root root 4.4K  Mar 14 22:46  install.log.syslog
-rw------- 1 root root 9.1K  Mar 25 00:21  mbox
-rw-r--r-- 1 root root 29K   Mar 14 22:53  install.log
```

介绍了这么多 ls 命令的选项，下面对 ls 命令进行一个总结。

【语法】**ls [选项] [目标文件或目录]**

选项说明：

-l：以长格式显示目录和文件信息。

-a：显示隐藏文件或目录。

-h：以易读的方式显示文件和目录的容量信息。

-d：显示目录本身的信息。

-t：按时间（由新到旧）对输出结果进行排序。

-S：按容量（由大到小）对输出结果进行排序。

-r：将排序结果反向输出。

ls 命令有很多参数，Linux 中的其他命令也有很多参数。对于这些参数，应该记住哪个？应该重点掌握哪个？这需要在工作中不断积累和摸索。在学习 Linux 之初，建议读者完整掌握本书所介绍的所有命令及其参数，这对于快速掌握 Linux 基本操作和继续下一阶段的学习是大有裨益的。

6. 切换工作目录

当需要切换工作目录时可以使用 cd 命令。

【语法】**cd 相对路径|绝对路径**

【示例】将工作目录切换到/boot/grub 目录中。

```
# cd  /boot/grub
```

对于 cd 命令还包含有几个特殊用法需要注意。

cd 不使用任何参数，进入当前用户的用户主目录。

cd ~ 进入当前用户的主目录。

cd ~用户名 进入特定用户的主目录。

cd .. 进入上级目录。

cd - 返回上一个工作目录。

当在不同的目录之间来回切换之时，如何判断当前的工作目录是什么呢？这里需要使用 pwd 命令，该命令用于显示当前工作目录的名称。例如：

```
[root@hero kernels]# pwd
/usr/src/kernels
```

7. 查看文件内容

（1）cat 命令

cat 命令用于直接查看文件内容。

【语法】**cat [选项] 文件名**

【示例】查看/etc/passwd 文件内容。

```
[root@hero ~]# cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
⋮
```

在使用这个命令时，如果文件很大，超过一个屏幕的显示容量时，cat 命令实现的是滚屏输出，这样用户就无法查看文件头部的信息，只能看到文件尾部最后一屏的内容。这是使用 cat 命令查看大文件时的一个缺点。可以采用其他查看方法来解决长文件的显示问题。在介绍其他查看方法之前，先介绍 cat 命令的选项，毕竟 cat 命令在查看小文件时使用起来还是很方便的。

【语法】**cat [选项] 文件名**

选项说明：

-n：查看文件时为文件的所有行添加行号。

-b：查看文件时为文件的非空白行添加行号。

-E：查看文件内容并显示行尾的换行符。

-T：查看文件内容，并将文件中的 Tab（制表位）用“^”来代替。

【示例】查看/etc/passwd 文件，并显示文件的行号。

```
[root@hero ~]# cat -n /etc/passwd
     1	root:x:0:0:root:/root:/bin/bash
     2	bin:x:1:1:bin:/bin:/sbin/nologin
     3	daemon:x:2:2:daemon:/sbin:/sbin/nologin
     4	adm:x:3:4:adm:/var/adm:/sbin/nologin
     5	lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
     6	sync:x:5:0:sync:/sbin:/bin/sync
     7	shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
     8	halt:x:7:0:halt:/sbin:/sbin/halt
     9	mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
    10	news:x:9:13:news:/etc/news:
    11	uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin
    12	operator:x:11:0:operator:/root:/sbin/nologin
     ⋮
```

【示例】查看/etc/passwd 文件，并显示文件的行号和结尾的换行符。

```
[root@hero ~]# cat -En /etc/passwd
     1   root:x:0:0:root:/root:/bin/bash$        ←$为 Linux 系统中的换行符
     2   bin:x:1:1:bin:/bin:/sbin/nologin$
     3   daemon:x:2:2:daemon:/sbin:/sbin/nologin$
     4   adm:x:3:4:adm:/var/adm:/sbin/nologin$
     5   lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin$
     6   sync:x:5:0:sync:/sbin:/bin/sync$
     7   shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown$
     8   halt:x:7:0:halt:/sbin:/sbin/halt$
     9   mail:x:8:12:mail:/var/spool/mail:/sbin/nologin$
    …
```

（2）more 命令

more 命令用于以翻页的形式查看文件的内容。

【语法】**more　文件名**

【示例】查看/etc/passwd 文件的内容。

```
[root@hero ~]# more /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
…
nscd:x:28:28:NSCD Daemon:/:/sbin/nologin
vcsa:x:69:69:virtual console memory owner:/dev:/sbin/nologin
rpc:x:32:32:Portmapper RPC user:/:/sbin/nologin
apache:x:48:48:Apache:/var/www:/sbin/nologin
mailnull:x:47:47::/var/spool/mqueue:/sbin/nologin
smmsp:x:51:51::/var/spool/mqueue:/sbin/nologin
pcap:x:77:77::/var/arpwatch:/sbin/nologin
--More--(51%)
```

注意，more 命令从第一行开始显示，当显示满一屏时，more 命令停止了显示，等待用户查看和继续操作。当用户查看完这一屏的内容后可以按空格键向下翻一页，或按回车键向下翻一行。在 more 中还可以使用“:f”命令显示当前文件名和光标所在的行数。

```
[root@hero ~]# more /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
…
smmsp:x:51:51::/var/spool/mqueue:/sbin/nologin
pcap:x:77:77::/var/arpwatch:/sbin/nologin
"/etc/passwd" line 23          ←使用":f"命令显示当前文件名和光标所在的行数
```

当文件内容被翻页显示到最底端时将自动退出 more 命令，如果不需要继续查看文件内容时可按 Q 键直接退出 more 命令。

more 这个文件查看程序与 cat 相比，其优点在于允许用户逐屏查看文件内容，对一些大文件而言这种查看方式是很方便的。但 more 命令也有其不足，那就是只能向下查看文件的内容，不能向上查看。例如，当一个用户向下翻屏后需要查看上一屏的内容时，more 命令就无能为力了。这时我们需要的是另一个命令 less。

注意，在新版本的 more 命令中增加了“b”向上翻页的功能。

（3）less 命令

less 命令用于以上下浏览的方式查看文件的内容。

【语法】**less 文件名**

【示例】以上下浏览的方式查看/etc/passwd 文件。

```
# less  /etc/passwd
```

在显示文件内容的过程中，可以按空格键向下翻一页，按 Page Down 向下翻一页，按 Page Up 向上翻一页。利用这些翻页键即可实现对文件的上下浏览式查看了。

同时 less 程序也提供字符串查找功能，使用方法是：输入“/字符串”表示向下查找字符串，输入“？字符串”表示向上查找字符串，n 为重复上一个查找，N 为反向重复上一查找。这一点与 man 中的字符串查找类似。

当完成文件的查看操作之后可以按 Q 键退出 less。

（4）tac 反向查看文件内容

tac 命令用于反向查看文件的内容，即从文件的最后一行开始显示文件的内容。这种显示方法在今后的日志分析以及脚本控制中的用途是很广泛的。

【语法】**tac 文件名**

【示例】反向查看/boot/grub/grub.conf 文件。

```
[root@hero ~]# tac /boot/grub/grub.conf
   initrd /initrd-2.6.18-194.el5.img
   kernel /vmlinuz-2.6.18-194.el5 ro root=/dev/VolGroup00/LogVol00 rhgb
quiet
   root (hd0,0)
title CentOS (2.6.18-194.el5)
hiddenmenu
splashimage=(hd0,0)/grub/splash.xpm.gz
timeout=5
default=0
#boot=/dev/hda
#          initrd /initrd-version.img
#          kernel /vmlinuz-version ro root=/dev/VolGroup00/LogVol00
#          root (hd0,0)
#          all kernel and initrd paths are relative to /boot/, eg.
# NOTICE:  You have a /boot partition.  This means that
```

```
# Note that you do not have to rerun grub after making changes to this file
#
# grub.conf generated by anaconda
```

（5）nl 命令

nl 命令用于显示文件内容并为文件添加行号。这个命令与“cat -n”命令类似，但不同于“cat -n”，nl 命令不仅可以添加行号，还可以控制行号添加的位置。

【语法】nl [选项] 文件名

【示例】显示/boot/grub/grub.conf 文件内容，并为文件添加行号。

```
[root@hero ~]# nl /boot/grub/grub.conf
     1  # grub.conf generated by anaconda
     2  #
     3  # Note that you do not have to rerun grub after making changes to
this file
     4  # NOTICE:   You have a /boot partition.  This means that
     5  #     all kernel and initrd paths are relative to /boot/, eg.
     6  #     root (hd0,0)
     7  #     kernel /vmlinuz-version ro root=/dev/VolGroup00/LogVol00
     8  #     initrd /initrd-version.img
     9  #boot=/dev/had

    10  default=0
    11  timeout=5
    12  splashimage=(hd0,0)/grub/splash.xpm.gz
    13  hiddenmenu
    14  title CentOS (2.6.18-194.el5)
    15   root (hd0,0)
    16   kernel/vmlinuz-2.6.18-14.el5 ro root=/dev/Voup00/Log00 rhgb quiet
    17   initrd /initrd-2.6.18-194.el5.img
```

通过输出可见，nl 为每一行均添加了行号，但是请注意，nl 并没有为空行添加行号。那么如果需要为空行也添加行号该如何处理呢？这里需要使用 nl 的-b 选项。

【语法】nl -b [a|t] 文件名

-b 选项定义了 nl 命令对空行的处理方式，其中：

-b a 表示不论是否为空行均添加行号。

-b t 表示如果有空行，则空行不添加行号，这是 nl 的默认值。

【示例】显示/boot/grub/grub.conf 文件内容，并为文件添加行号，如果文件中包含空行，则为空行添加行号。

```
[root@hero ~]# nl -b a /boot/grub/grub.conf
     1  # grub.conf generated by anaconda
     2  #
     3  # Note that you do not have to rerun grub after making changes to
```

```
        this file
     4  # NOTICE:  You have a /boot partition.  This means that
     5  #          all kernel and initrd paths are relative to /boot/, eg.
     6  #          root (hd0,0)
     7  #          kernel /vmlinuz-version ro root=/dev/VolGroup00/
                   LogVol00
     8  #          initrd /initrd-version.img
     9  #boot=/dev/hda
    10 ←空行被添加行号
    11  default=0
    12  timeout=5
    13  splashimage=(hd0,0)/grub/splash.xpm.gz
    14  hiddenmenu
    15  ←空行被添加行号
    16  title CentOS (2.6.18-194.el5)
    17          root (hd0,0)
    18          kernel /vmlinuz-2.6.18-194.el5 ro root=/dev/VolGroup00/
                LogVol00 rhgb quiet
    19          initrd /initrd-2.6.18-194.el5.img
```

nl 命令除了可以控制是否为空行添加行号之外，还可以控制行号添加的位置。当需要控制行号添加的位置时，可以使用-n 选项。

【语法】**nl　-n [ln|rn|rz]　文件名**

-n　ln：行号在屏幕的最左方显示。

-n　rn：行号在字段头部的右侧显示，且不添 0 补齐空白处。

-n　rz：行号在字段头部的右侧显示，且添 0 补齐空白处。

对于上述参数的功能用以下几个示例来说明。

【示例】显示/boot/grub/grub.conf 文件内容，并为文件添加行号，行号在屏幕的最左方显示。

```
[root@hero ~]# nl -n ln /boot/grub/grub.conf
1       # grub.conf generated by anaconda
2       #
3       # Note that you do not have to rerun grub after making changes to
          this file
4       # NOTICE:You have a /boot partition.  This means that
5       #          all kernel and initrd paths are relative to /boot/, eg.
6       #          root (hd0,0)
7       #              kernel /vmlinuz-version ro root=/dev/VolGroup00/
                       LogVol00
8       #              initrd /initrd-version.img
9       #boot=/dev/had
…
```

【示例】显示/boot/grub/grub.conf 文件内容，并为文件添加行号，行号在屏幕的最左

侧显示。

```
[root@hero ~]# nl -n rn /boot/grub/grub.conf
     1  # grub.conf generated by anaconda
     2  #
     3  # Note that you do not have to rerun grub after making changes to
          this file
     4  # NOTICE:    You have a /boot partition.  This means that
     5  #            all kernel and initrd paths are relative to
                     /boot/, eg.
     6  #            root (hd0,0)
     7  #            kernel /vmlinuz-version ro root=/dev/VolGroup00/
                     LogVol00
     8  #            initrd /initrd-version.img
   …
```

【示例】显示/boot/grub/grub.conf 文件内容，并为文件添加行号，行号在屏幕的最左侧显示，空白处添 0 补齐。

```
[root@hero ~]# nl -n rz /boot/grub/grub.conf
000001  # grub.conf generated by anaconda
000002  #
000003  # Note that you do not have to rerun grub after making changes to
this file
000004  # NOTICE:    You have a /boot partition.  This means that
000005  #            all kernel and initrd paths are relative to /boot/,
                     eg.
000006  #            root (hd0,0)
000007  #            kernel /vmlinuz-version ro root=/dev/VolGroup00/
                     LogVol00
000008  #            initrd /initrd-version.img
000009  #boot=/dev/had
…
```

在上例中可见，行号共占用 6 位字符，而行号实际最大为 17，即仅使用了两位空间。这样不仅使得输出格式发生了较大的偏移，而且可读性也不高。使用 nl 命令的-w 选项还可以控制行号所占用的位数。

【语法】**# nl -w 位数 文件名**

【示例】显示/boot/grub/grub.conf 文件内容，并为文件添加行号，行号在屏幕的最左侧显示，且行号仅占用两位。

```
[root@hero ~]# nl -n rz -w 2 /boot/grub/grub.conf
01      # grub.conf generated by anaconda
02      #
03      # Note that you do not have to rerun grub after making changes to
this file
```

```
04      # NOTICE:   You have a /boot partition.  This means that
05      #           all kernel and initrd paths are relative to /boot/,
                    eg.
06      #           root (hd0,0)
07     #            kernel /vmlinuz-version ro root=/dev/VolGroup00/
                    LogVol00
08      #           initrd /initrd-version.img
09      #boot=/dev/had
…
```

在介绍完 nl 的使用方法后，下面总结一下 nl 的语法结构。

【语法】nl [选项] 文件名

选项说明：

-b 选项：对空行行号的处理，其中：

-b a：表示不论是否为空行均添加行号；

-b t：表示如果有空行，空行不添加行号。

-n 选项：对行号位置的处理，其中：

-n ln：行号在屏幕的最左侧显示；

-n rn：行号在字段头部的右侧显示，且不添 0 补齐空白处；

-n rz：行号在字段头部的右侧显示，且添 0 补齐空白处。

-w 位数：定义行号所占用的位数。

以上的命令均是对整个文件内容的查看，但在实际工作中经常会遇到仅查看文件首部或文件尾部的情况。如配置文件的说明信息常常在文件的首部，日志文件的更新记录常常在文件的尾部。为了能方便地查看文件的首部和尾部的信息，可以使用 head 和 tail 两个命令。

（6）head 命令

head 命令用于显示文件首部的内容，默认的情况下将显示文件前 10 行的内容。

【语法】# head 文件名

【示例】 显示/boot/grub/grub.conf 文件的首部内容。

```
[root@hero ~]# head /boot/grub/grub.conf
# grub.conf generated by anaconda
#
# Note that you do not have to rerun grub after making changes to this file
# NOTICE:   You have a /boot partition.  This means that
#           all kernel and initrd paths are relative to /boot/, eg.
#           root (hd0,0)
#           kernel /vmlinuz-version ro root=/dev/VolGroup00/LogVol00
#           initrd /initrd-version.img
#boot=/dev/hda
```

注意，head 命令默认仅显示文件的前 10 行内容，如果要改变显示的行数，需要使

用-n 选项。

【语法】**# head　-n　行数　文件名**

选项说明：

-n 选项用于定义 head 命令显示的行数。

【示例】显示/boot/grub/grub.conf 文件的前 15 行内容。

```
[root@hero ~]# head -n 15 /boot/grub/grub.conf
# grub.conf generated by anaconda
#
# Note that you do not have to rerun grub after making changes to this file
# NOTICE:   You have a /boot partition.  This means that
#           all kernel and initrd paths are relative to /boot/, eg.
#           root (hd0,0)
#           kernel /vmlinuz-version ro root=/dev/VolGroup00/LogVol00
#           initrd /initrd-version.img
#boot=/dev/hda

default=0
timeout=5
splashimage=(hd0,0)/grub/splash.xpm.gz
hiddenmenu
```

（7）tail 命令

tail 命令与 head 命令的功能正好相对，用于显示文件尾部的内容，默认为显示文件尾部 10 行的内容。

【语法】**# tail　文件名**

【示例】显示/etc/passwd 文件的末尾内容。

```
[root@hero ~]# tail /etc/passwd
gdm:x:42:42::/var/gdm:/sbin/nologin
user1:x:500:500::/home/user1:/bin/bash
user2:x:501:501::/home/user2:/bin/bash
user3:x:502:502::/home/user3:/bin/bash
user4:x:503:503::/home/user4:/bin/bash
user5:x:504:504::/home/user5:/bin/bash
user6:x:505:505::/home/user6:/bin/bash
user7:x:506:506::/home/user7:/bin/bash
user8:x:507:507::/home/user8:/bin/bash
user9:x:508:508::/home/user9:/bin/bash
```

注意，tail 命令默认仅显示文件末尾 10 行内容。与 head 命令一样，如果要改变显示的行数，需要使用-n 选项。

【语法】**# tail　-n　行数　文件名**

【示例】显示/etc/passwd 文件的末尾 4 行的内容。

```
[root@hero ~]# tail -n 4 /etc/passwd
user6:x:505:505::/home/user6:/bin/bash
user7:x:506:506::/home/user7:/bin/bash
user8:x:507:507::/home/user8:/bin/bash
user9:x:508:508::/home/user9:/bin/bash
```

8. 文件与目录处理

1）创建目录

创建目录使用 mkdir 命令，该命令的语法结构如下：

【语法】**# mkdir [-p] 目录名**

选项说明：

p：用于递归创建目录，即如果被创建的目录的父目录不存在，则一并创建父目录。

【示例】在当前用户主目录下创建名为 xinya 的目录。

```
[root@hero etc]# pwd
/etc
[root@hero etc]# cd ~
[root@hero ~]# mkdir xinya
[root@hero ~]# ls
anaconda-ks.cfg  Desktop  file1  install.log  install.log.syslog  mbox
test  xinya
```

在创建目录时要注意，被创建的目录的父目录要事先存在。例如，要创建~/xinya/lpi/test1/这个目录，要求~/xinya/lpi/这个目录要事先存在，如果不存在，系统将提示“不能创建该目录，没有该文件或目录”的错误信息。例如：

```
[root@hero ~]# pwd
/root
[root@hero ~]# ls
anaconda-ks.cfg  Desktop  file1  install.log  install.log.syslog  mbox
test  xinya
[root@hero ~]# mkdir /root/lpi/xinya
mkdir: cannot create directory '/root/lpi/xinya': No such file or directory
```

mkdir 命令提供了-p 选项，该选项用于递归创建目录，即如果被创建的目录的父目录不存在，则一并创建父目录。

【语法】**# mkdir -p 目录名**

【示例】创建~/xinya/lpi/test1/目录。

```
[root@hero ~]# pwd
/root
[root@hero ~]# ls
Desktop  anaconda-ks.cfg  file1  install.log  install.log.syslog  mbox
test  xinya
```

```
[root@hero ~]# mkdir ~/lpi/test1
mkdir: cannot create directory '/root/lpi/test1': No such file or directory
[root@hero ~]# mkdir -p ~/lpi/test1
[root@hero ~]# ls
Desktop  anaconda-ks.cfg  file1  install.log  install.log.syslog  lpi
mbox  test  xinya
```

2）删除目录

在系统管理中删除目录存在两种情况：一种情况是删除空目录，即目录中没有文件；另一种情况是删除非空目录，即目录中有文件。

（1）删除空目录可以使用 rmdir 命令，该命令仅用于删除空目录。

【语法】**# rmdir [-p] 目录名**

选项说明：

p：用于递归删除。即当删除目录时，如果父目录为空则一并删除。

【示例】删除~/lpi/xinya/这个空目录。

```
[root@hero ~]# ls ~/lpi/
xinya
[root@hero ~]# rmdir ~/lpi/xinya
[root@hero ~]# ls ~/lpi/
```

rmdir 也提供了-p 选项，用于递归删除。即当删除目录时，如果父目录为空则一并删除。例如在~xinya/lpi/目录中，使用“rmdir -p xinya/lpi”命令删除 lpi 目录，如果 xinya 目录中不再有其他目录，则 xinya 目录也会被一并删除，这实际上是一种递归删除。

```
[root@hero ~]# ls
Desktop  anaconda-ks.cfg  file1  install.log  install.log.syslog  lpi
mbox  test  xinya
[root@hero ~]# ls ~/xinya/
lpi
[root@hero ~]# rmdir -p xinya/lpi
[root@hero ~]# ls
Desktop  anaconda-ks.cfg  file1  install.log  install.log.syslog  lpi
mbox  test
```

（2）删除非空目录可以使用 rm -r 命令，该命令用于删除非空目录。使用 rm -r 删除非空目录是一个递归删除的过程，即先删除目录下的文件，然后再删除目录本身。

【语法】**# rm -r 目录名**

【示例】删除包含文件的 xinya 目录。

```
[root@hero ~]# rm -r xinya
rm: descend into directory 'xinya'? y
rm: remove regular empty file 'xinya/file2'? y      ←首先删除目录中的文件
rm: remove regular empty file 'xinya/file3'? y
rm: remove regular empty file 'xinya/file1'? y
```

```
rm: remove directory 'xinya'? y          ←删除目录中的文件后,再删除目录本身
```

当使用“rm -r”命令删除非空目录时，先要进入目录删除文件，而且在每删除一个文件前将询问用户是否确认删除，用户按 y 键表示确认删除，按 n 键表示不删除。当不同意删除文件时，文件将被保留在该目录中，且该目录不会被删除。

当一个目录中的文件很少时，这种交互式操作的问题不大，但当一个目录中包含几百个甚至几千个文件时，删除该目录将回答几千次 y 将变得很麻烦，这时可使用-f 选项，-f 选项的作用是强制删除，不需要经过用户确认。

还是上面的操作，当使用“rm -rf”选项时将直接删除目录而无需用户确认。

```
[root@hero ~]# rm -rf xiya
```

【语法】# rm -rf 目录名

选项说明：

r：用于删除非空目录。

f：表示强制删除，不需要经过用户确认。

3）创建文件

在 Linux 中创建文件有多种方法，下面介绍一种最直观的方法，使用 touch 命令。

【语法】# touch 文件名

touch 命令的作用表现在两个方面。一方面是当目标文件已存在时，touch 命令用于更新文件的时间标识信息，即将文件的时间戳信息更新为当前的系统时间。另一方面，当目标文件不存在时，touch 命令用于创建目标文件。下面通过两个示例来说明 touch 命令在这两方面的作用。

【示例】当前目录下已存在 file1 这个文件，使用 touch 命令更新 file1 的时间标识信息。

```
[root@hero ~]# touch file1
[root@hero ~]# ll file1
-rw-r--r-- 1 root root 12866 Mar 29 12:10 file1
```

【示例】在当前目录下创建 file2 文件。

```
[root@hero ~]# ls
Desktop  anaconda-ks.cfg  file1  install.log  install.log.syslog  lpi
mbox  test  xinya
[root@hero ~]# touch file2
[root@hero ~]# ll file2
-rw-r--r-- 1 root root 0 Mar 29 12:11 file2
```

4）删除文件

删除文件可以使用 rm 命令。

【语法】# rm 文件名

【示例】删除 xinya 目录下的 file1 文件。

```
[root@hero ~]# ll xinya
total 0
-rw-r--r-- 1 root root 0 Mar 29 12:09 file1
-rw-r--r-- 1 root root 0 Mar 29 12:09 file2
-rw-r--r-- 1 root root 0 Mar 29 12:09 file3
[root@hero ~]# rm xinya/file1
rm: remove regular empty file 'xinya/file1'? y
[root@hero ~]# ll xinya
total 0
-rw-r--r-- 1 root root 0 Mar 29 12:09 file2
-rw-r--r-- 1 root root 0 Mar 29 12:09 file3
```

利用 rm 命令删除文件时，系统会询问用户是否真的要删除该文件，按 y 键（yes）表示删除，按 n 键（no）表示不删除。这样做的目的是防止误删除文件。

rm 命令可以同时删除多个文件，如“rm　file2 file3 file4”，文件名间用空格隔开。

使用 rm 命令时，由于每删除一个文件系统均要求用户确认，这里删除了 4 个文件，所以进行了 4 次确认。如果想在删除的过程中不需要用户确认，可以使用 rm 命令的-f 选项。-f 选项的作用是强制删除，不要求用户确认。

rm 命令在前面删除非空目录时就已经接触并使用了，下面将这个命令的语法做一个总结。

【语法】**# rm　[-r f]　文件或目录名**

选项说明：

r：用于递归删除目录。

f：用于强制删除文件或目录。

5）文件的复制与移动

（1）文件的复制

复制文件可以使用 cp 命令，该命令的作用是复制源文件到目标文件，或复制多个源到目标。

【语法】**# cp　源文件　目标文件**

【示例】将/boot/grub/grub.conf 文件复制到/root/xinya 文件夹内，名称是 grub.conf.bak。

```
[root@hero ~]# ll /boot/grub/grub.conf
-rw------- 1 root root 607 Mar 29 10:46 /boot/grub/grub.conf
[root@hero ~]# cp /boot/grub/grub.conf  /root/xinya/grub.conf.bak
[root@hero ~]# ll /root/xinya/grub.conf.bak
-rw------- 1 root root 607 Mar 29 12:14 /root/xinya/grub.conf.bak
```

这里将/boot/grub/grub.conf 作为源文件，将/root/xinya/grub.conf.bak 作为目标文件，实现了将源文件的内容复制到目标文件的目的。在这个复制过程中，目标文件的创建时间是当前时间，即目标文件并未继承源文件的时间属性。

在这个例子中目标文件事先不存在，而是在复制时创建生成的，即如果目标文件不存在则创建该文件。如果目标文件事先已存在，则系统会询问用户是否覆盖原文件。如

/root/xinya 文件夹内存在一个 grub.conf.bak 文件。当需要将/boot/grub/grub.conf 文件复制到/root/backup/目录下并起名叫 grub.conf.backup 文件时，系统会询问用户是否用新内容覆盖原有的内容。

```
[root@hero ~]# ll /root/xinya/grub.conf.bak
-rw------- 1 root root 607 Mar 29 12:14 /root/xinya/grub.conf.bak
[root@hero ~]# cp /boot/grub/grub.conf  /root/xinya/grub.conf.bak
cp: overwrite '/root/xinya/grub.conf.bak'? y
```

在这个例子中，系统询问用户是否覆盖/root/backup/grub.conf.backup，按 y 键（yes）表示同意覆盖，按 n 键（no）表示不同意覆盖。

在复制的过程中如果没有指明目标文件名，则目标文件名与源文件名相同。如“cp /boot/grub/grub.conf　/root/xinya/”，这里没有定义目标文件名，则目标文件名与源文件相同。

```
[root@hero ~]# ll /root/xinya/
total 4
-rw-r--r-- 1 root root   0 Mar 29 12:09 file3
-rw------- 1 root root 607 Mar 29 12:15 grub.conf.bak
[root@hero ~]# cp /boot/grub/grub.conf  /root/xinya/
[root@hero ~]# ll /root/xinya/
total 8
-rw-r--r-- 1 root root   0 Mar 29 12:09 file3
-rw------- 1 root root 607 Mar 29 12:15 grub.conf
-rw------- 1 root root 607 Mar 29 12:15 grub.conf.bak
```

本例中并未指明目标文件名，这时目标文件名与源文件名相同。

在复制过程中如果希望保留源文件的时间标记，即使用与源文件相同的创建时间，可以使用 cp 命令的-p 选项，该参数的作用是保留源文件属性。

```
[root@hero ~]# ll /boot/grub/grub.conf
-rw------- 1 root root 607 Mar 29 10:46 /boot/grub/grub.conf
[root@hero ~]# cp -p /boot/grub/grub.conf /root/xinya/grub
[root@hero ~]# ll /root/xinya/grub
-rw------- 1 root root 607 Mar 29 10:46 /root/xinya/grub
```

cp 命令除文件复制功能外还可以复制目录。采用-a 选项将实现复制目录及目录下的所有子目录与文件，并保留目录的原属性。

【示例】将/var/log/目录复制到/root/目录中。

```
# cp -a /var/log  /root/log
```

本例中/var/log 为源目录，/root/log 为目标目录。当采用-a 选项后，cp 命令将/var/log/目录中的所有文件和子目录均复制到了/root/log/目录中，并且保留了/var/log/目录的相关属性。cp 这个命令的-a 选项相当于 cp -dpR。cp 命令的各个选项及其功能见表 3.4。

表 3.4　cp 命令的选项

选 项 名 称	选 项 功 能
-d	复制符号连接
-p	保留原文件和目录的属性信息
-R	处理指定目录及目录下的子目录下的所有文件
-a	相当于 dpR

如果所复制的目录不需要保留源目录的属性，可以采用 cp -dR 命令来实现。

cp 命令还可实现将多个文件移动到一个目录的操作。

【语法】**# cp　文件 1　文件 2　文件 3　目标目录**

【示例】将/root/目录下的 install.log、install.log.syslog 和 anaconda-ks.cfg 这 3 个文件同时移动到/root/xinya/目录中。

```
# cp  install.log  install.log.syslog  anaconda-ks.cfg  xinya/
```

（2）文件的移动

文件的移动可以使用 mv 命令。

【语法】**# mv　源文件　目标文件**

mv 命令的作用是将源文件移动到目标文件处。

【示例】将/root/目录中的 telnum 文件移动到/root/xinya/目录中。

```
# mv  /root/telnum  /root/xinya/telnum
```

这里/root/telnum 作为源文件被移动到了/root/xinya/目录中并取名为 telnum，而/root/目录下将不存在 telnum 文件。

在本例中指明了目标文件的文件名。如果没有指明目标文件的文件名，则目标文件采用源文件的文件名。与 cp 命令相同，如果目标文件事先已经存在，则系统会询问用户是否覆盖原文件。

如果源文件与目标文件处于相同目录下，则 mv 命令实现的是文件重命名。

【示例】将/root/xinya/telnum 文件重命名为 linuxinfo.txt。

```
# mv  /root/xinya/telnum  /root/xinya /linuxinfo.txt
```

mv 命令还可实现将多个文件移动到一个目录中。如将/root/目录中的 file1、file2、file3 和 file4 移动到/root/dir1 目录中，可以使用 mv 命令来实现。

【示例】# mv　file1　file2　file3　file4　/root/dir1/

6）文件查找

Linux 中文件的查找方式比较多，下面介绍 which、whereis、locate 以及 find 这几种搜索命令的使用方法。

（1）查找可执行文件——which 命令

which 命令用于查找可执行文件。该命令会以 PATH 环境变量所定义的路径作为查

找路径。所以，如果一个可执行文件的路径未包含在 PATH 环境变量设定的路径中，则无法查找到。由于 PATH 环境变量默认设定了系统中所有命令的存放路径，所以 which 命令默认的情况下只用于查找系统命令的存放位置。

which 命令的语法格式是“which 文件名”。如查找 passwd 命令的位置，可用“which passwd”命令来实现。

```
[root@hero ~]# which passwd
/usr/bin/passwd
```

【语法】**which [-a] 命令名**

选项说明：

a：显示查找到的所有匹配的可执行文件路径。

（2）查找文件或目录——whereis 命令

whereis 命令用于查找文件或目录。该命令仅对特定目录下的文件或目录进行查找，所以其查找结果不能反映文件在系统中存在的完全信息。

whereis 命令的语法格式是“whereis 字符串”，查找与字符串匹配的文件或目录名称。如“whereis passwd”查找包含 passwd 字符串的文件。

```
[root@hero ~]# whereis passwd
passwd:  /usr/bin/passwd  /etc/passwd  /usr/share/man/man1/passwd.1.gz
/usr/share/man/man5/passwd.5.gz
```

whereis 命令列出了所有包含有 passwd 字符串的文件名和目录。如果使用 whereis 命令只想得到可执行文件，可利用 whereis 命令的-b 选项。该选项的作用是只寻找二进制可执行文件。如“whereis -b passwd”查找包含 passwd 字符串的可执行文件。

```
[root@hero ~]# whereis -b passwd
passwd: /usr/bin/passwd /etc/passwd
```

如果只想查找帮助文件，则可使用 whereis 的-m 选项。该选项的作用在于仅在 man 帮助文件中查找。如“whereis -m passwd”查找包含 passwd 字符串的帮助文件。

```
[root@hero ~]# whereis -m passwd
passwd: /usr/share/man/man1/passwd.1.gz /usr/share/man/man5/passwd.5.gz
```

【语法】**whereis [选项] 字符串**

选项说明：

b：仅查找二进制文件。

m：查找帮助文件。

s：查找源文件。

u：查找其他类型文件。

（3）查找文件或目录——locate 命令

locate 命令用于查找文件或目录。该命令的查找机制是利用文件列表数据库来实现

对文件或目录的查找。Linux 会将系统中所有的文件和目录名称及位置记录在一个文件列表数据库中，当执行 locate 命令时，会在该数据库内查找文件，并将文件的位置信息显示给用户。该数据库每天 5:50 分自动更新一次，如果数据库还未更新且文件结构发生了变化，如添加或删除了某些文件，则这些变化将不能记入数据库。所以当利用 locate 命令查找文件时会发生文件无法找到或找到已经删除的文件的情况。解决这个问题的方法是使用 updatedb 命令立即更新数据库，使得数据库能正确反映文件系统的状态。该数据库的位置在/var/lib/slocate/slocate.db。由于以数据库为依托，所以 locate 命令的查询效率要高于 whereis。

locate 命令的语法格式为：

【语法】**# locate 字符串**

【示例】使用 locate passwd 查找包含 passwd 字符串的文件或目录名。

```
[root@hero ~]# locate passwd
/etc/passwd
/etc/passwd-
/etc/pam.d/passwd
/etc/security/opasswd
/lib/security/pam_passwdqc.so
/lib/security/pam_unix_passwd.so
/usr/bin/RSA_SecurID_getpasswd
/usr/bin/gpasswd
/usr/bin/htpasswd
/usr/bin/kdepasswd
/usr/bin/lppasswd
/usr/bin/passwd
…
```

（4）查找文件或目录——find 命令

find 是一个功能强大的文件或目录查找命令。该命令具有查找方法灵活、查找内容全面的优点。但是该命令也有磨损硬盘和查找速度慢的缺点。推荐大家在利用 whereis 和 locate 命令不能定位文件或目录时再使用 find 命令。

【语法】**# find [查找的起始路径] [选项] 字符串**

【示例】由根目录开始查找 passwd 文件。

```
[root@hero ~]# find / -name passwd
/etc/pam.d/passwd
/etc/passwd
/usr/bin/passwd
/usr/share/doc/nss_ldap-253/pam.d/passwd
```

-name 选项的作用是指明待查找的文件名是 passwd。

7）查看文件类型

Linux 中的文件类型与 Windows 不同。Windows 利用扩展名来标识文件类型并关联

应用程序，而 Linux 系统中文件的扩展名不具有该功能。那么在 Linux 系统中如何识别文件的类型呢？可以使用 file 命令。

file 命令的作用在于识别文件类型。

【语法】**# file 文件名**

【示例】查看/etc/passwd 文件的类型。

```
# file /etc/passwd
[root@hero ~]# file /etc/passwd
/etc/passwd: ASCII text
```

可见，/etc/passwd 文件是一个 ASCII 码格式的文本文件。

file 命令用于判断文件的类型，能够识别的文件类型有目录、Shell Script、ASCII 文本、二进制可执行文件（ELF 格式）、C 语言源文件、图形文件以及压缩文件。

本节内容比较多，主要目的是让大家掌握 Linux 的基本操作。本节用较大的篇幅来介绍常见命令的使用方法，是考虑到初学者对命令的使用比较陌生，同时需要培养命令的使用习惯，所以尽可能详细地解释每个命令与选项的含义。限于篇幅，在读者养成良好的命令使用习惯之后，后面各章对命令的介绍将趋于简明扼要。

第4章 文本编辑工具 vim

Linux 系统中各种功能的设置以及系统、环境等设置通常存放在特定的配置文件内，这些配置文件大部分是纯文本文件。在使用 Linux 系统时常常需要修改一些系统、环境或功能的设置，所以编辑文本就成为一项必要的工作。熟悉 MS Windows 的用户大都使用过 Notepad（记事本）来编辑过文件，在 Linux 系统内也有一些基本的文本编辑工具。

本章主要介绍文本编辑工具 vim 的使用。vim 是当前 UNIX 操作系统和 Linux 操作系统中最常见的文本编辑工具，它具有体积小、效率高、功能丰富、使用广泛等诸多优点，而且在一些发行版的系统恢复模式或单用户模式中，vim 是唯一可用的文本编辑器。所以学习使用 vim 编辑器对 Linux 系统的管理工作是非常重要的。

vim 编辑器是一种行式编辑器，与我们在 Windows 中常用的编辑器有所不同，读者在学习过程中要逐渐适应这种变化，以期充分理解和掌握这种编辑器的使用方法。

4.1 vi 编辑器

vim 编辑器是 Visual Interface 的简称，通常称之为 vim。它可以执行输出、删除、查找、替换和块操作等众多文本操作，而且用户可以根据自己的需要对其进行定制，这是其他编辑程序所没有的。

vim 编辑器并不是一个排版程序，它不像 Word 或 WPS 那样可以对字体、格式和段落等其他属性进行编排，它只是一个文本编辑程序。vim 没有菜单，只有命令，且命令繁多。

4.1.1 vi 与 vim

vim 的前身是 vi，当前大多数系统中均支持 vim。在一些发行版中，vi 已成为了 vim 的别名，使用 vi 就是使用 vim。

vim 较之 vi 而言增加了一系列的扩展功能，包括语法提示、多窗口和块选择等。

4.1.2 vim 的启动与模式介绍

1. vim 的启动与结束

启动 vi 使用 vi 命令或 vim 命令（注意，在一些发行版中，vi 与 vim 是相同的，都是启动 vim 编辑器，但在一些发行版中 vi 启动的是 vi 编辑器，vim 启动的是 vim 编辑器）。

【示例】启动 vi。

```
[root@hero ~]# vi
~
~                    VIM - Vi IMproved
~                      version 7.0.237
~                   by Bram Moolenaar et al.
~            Vim is open source and freely distributable
~                 Become a registered Vim user!
~           type  :help register<Enter>  for information
~           type  :q<Enter>              to exit
~           type  :help<Enter> or <F1>   for on-line help
~           type  :help version7<Enter>  for version info
```

在本例中，使用 vi 启动的就是 vim 编辑器。启动 vim 编辑器后可以看到有关 vim 编辑器的版本信息和相关的帮助信息。但在有些发行版中还需要使用 vim 命令来启动 vim 编辑器。

如果需要在启动 vim 的同时打开一个文本文件进行编辑，就需要加上文件名，例如要编辑 test 文件，输入：

```
# vim  test
```

或

```
# vi  test
```

将打开一个全屏行式编辑界面。注意，如果 vim 命令后的文件已经存在，则打开文件；如果 vim 命令后的文件不存在，则创建文件并打开。

在 vim 中编辑的文件都先存放在缓冲区中，任何改变也只会改变缓冲区中的内容，并不直接改变原来的文件内容。只有当用户下达保存文件的指令之后才真正改变文件的内容。

当编辑完成后，可以使用“:q”退出编辑工具。

2. vi 编辑器的模式

vi 编辑器存在 3 种模式，分别是命令模式、插入编辑模式和末行模式。这 3 种模式的关系如图 4.1 所示。

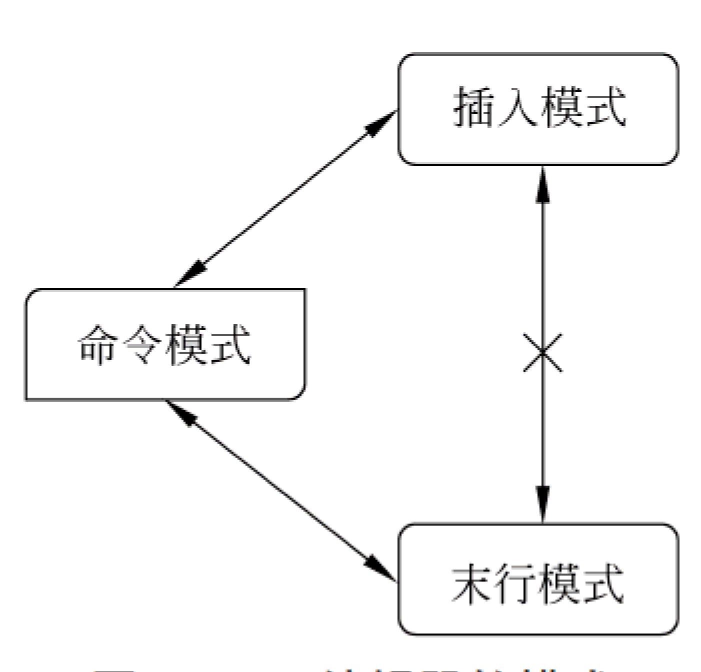


图 4.1　vi 编辑器的模式

当启动 vi 编辑器时，默认会进入命令模式，该模式下仅接受命令操作。当需要修改文件内容时，需要进入到插入编辑模式，在该模式下可以实现对文件内容的编辑。当需要执行保存、退出等功能性操作时需要进入末行模式，在该模式下使用末行命令可以实现对文件的一系列的功能性操作。

需要注意的是，在这 3 种模式之间，命令模式是其他

两种模式的中心转换模式。也就是说只能由命令模式进入插入模式，由插入模式进入命令模式；或者由命令模式进入末行模式，由末行模式进入命令模式。但不能由插入模式直接进入末行模式。

3. 模式之间的切换

1）命令模式与插入模式之间的切换

（1）由命令模式进入插入模式的指令见表 4.1。

表 4.1　vi 由命令模式进入插入模式的指令

指　　令	功　　能
i	在光标前插入字符
I	在光标所在行的行首插入字符
a	在光标后插入字符
A	在光标所在行的行尾插入字符
o	在光标所在行的下一行新建一行，在行首插入字符
O	在光标所在行的上一行新建一行，在行首插入字符
R	从光标的位置开始，以输入的字符覆盖原有的字符

（2）由插入模式进入命令模式用 Esc 键实现。

2）命令模式与末行模式之间的切换

（1）由命令模式进入末行模式使用“:末行命令”来实现。

（2）由末行模式进入命令模式，当末行命令执行完毕后会根据程序运行情况自动回到命令模式。

4.1.3　命令模式下的操作

vi 的命令模式下有很多操作，包括移动光标、删除文本、复制和粘贴、查找文本和查找替换等操作。接下来就逐一介绍这部分操作命令。本节的内容需要读者打开一个文件，对照本节内容来逐一实验，以便更好地掌握这一系列的功能按键。建议读者将 /etc/lftp.conf 文件复制到用户主目录，并使用该副本文件进行练习。

1. 移动光标的位置（见表 4.2）

表 4.2　vi 在命令模式下的移动光标命令

命　　令	功　　能
h	←向左移动光标，10h 表示光标向左移动 10 个字符
l	→向右移动光标，10l 表示光标向右移动 10 个字符
j	↓向下移动光标，10j 表示光标向下移动 10 行
k	↑向上移动光标，10k 表示光标向上移动 10 行
0	光标移至行首
$	光标移至行尾

续表

命　　令	功　　能
H	光标移至屏幕最上行行首
M	光标移至屏幕中间行行首
L	光标移至屏幕最下行行首
n G	光标移至第 *n* 行
G	光标移至文件的最后一行行首
1 G	光标移至文件的第一行行首，该命令与 gg 命令相同
n Enter	光标向下移动 *n* 行

2. 删除文本（见表 4.3）

表 4.3　vi 在命令模式下的删除文本命令

命　　令	功　　能
dd	删除光标所在行
*n*dd	删除由光标所在行开始的向下的 *n* 行
D	删除光标后的所有字符
u	还原上一次操作
x	向后删除一个字符，等同于 del 键
*n*x	向后删除 *n* 个字符
X	向前删除一个字符，等同于 Backspace 键
*n*X	向前删除 *n* 个字符
d1G	从当前行开始删除至文件的第一行
dG	从当前行开始删除至文件的尾行
d$	从光标当前位置开始删除至行尾
d0	从光标当前位置开始删除至行首

3. 复制与粘贴（见表 4.4）

表 4.4　vi 在命令模式下的复制与粘贴命令

命　　令	功　　能
yy	复制光标所在行
yw	复制光标位置到单字结束的字符
*n*yy	复制由光标所在行开始向下的 *n* 行
y1G	复制光标所在行到文件的第一行
yG	复制光标所在行到文件的最后一行
y0	复制光标当前位置到行首的内容
y$	复制光标当前位置到行尾的内容
p	将已复制的数据粘贴到光标的下一行
P	将已复制的数据粘贴到光标的上一行
J	将光标所在的当前行与下一行合并为一行

4．查找文本（见表 4.5）

表 4.5　vi 在命令模式下的查找文本命令

命　　令	功　　能
/字符串	向下查找字符串
/	重复上一次的向下查找
? 字符串	向上查找字符串
?	重复上一次的向上查找
n	重复上一个查找
N	重复上一个查找，但方向相反

5．查找替换

查找替换是 vim 中常用的一种操作。在 vim 中要实现查找替换操作需要使用“:s”命令。

【语法】**:s /字符串 1/字符串 2/**

该命令的作用是将光标所在行的第一个“字符串 1”替换为“字符串 2”。例如：

```
1  ## some useful aliases
2  alias dir ls
3  alias less more
4  alias zless zmore
5  alias bzless bzmore
6  alias reconnect "close; cache flush; cd ."
```

这是 lftp.conf 文件的前 6 行，当前光标停留在第一行。如果需要将该行中的“aliases”字符串替换为“aname”，这时可使用“:s/aliases/anme/”命令，该命令会将光标所在行的“aliases”字符串替换为“aname”字符串。

```
:s/aliases/aname/

## some useful aname
alias dir ls
alias less more
alias zless zmore
alias bzless bzmore
alias reconnect "close; cache flush; cd ."
```

从刚刚的替换过程可见，这种替换方法的效率不高，只能对光标所在行的第一个匹配字符串进行替换。其实，“:s”命令还有其他一系列的用法，用于实现高效的查找替换操作。

【语法】**:s /字符串 1/字符串 2/g**

该命令的作用是将光标所在行的所有“字符串 1”均替换为“字符串 2”。注意在这

个命令中的“/g”表示的是完全替换，而不是仅替换第一个。

【语法】**:a，bs/字符串 1/字符串 2/g**

该命令的作用是将文档中由第 a 行开始到第 b 行结束的所有的“字符串 1”替换为“字符串 2”，这里 a 表示的是起始行号，b 表示的是结束行号。

【示例】将 lftp.conf 文件中由第一行开始到第 10 行结束的所有“alias”字符串替换为“aname”字符串。

```
:1,10s/alias/anme/g
```

【语法】**:n，$s/字符串 1/字符串 2/g**

该命令的作用是将第 n 行开始到最后一行结束的所有“字符串 1”替换为“字符串 2”。注意，这里的“$”表示最后一行。

【语法】**:.,$s/字符串 1/字符串 2/gc**

该命令的作用是将由当前行开始到文件最后一行结束的所有“字符串 1”替换为“字符串 2”，并且在每进行一次替换时均要求用户确认。注意，这里的“.”表示当前目录，“c”表示用户确认。

4.1.4 末行模式

在末行模式下，可以实现诸如保存、退出以及另存为等一系列操作。

末行模式下的常用命令见表 4.6。

表 4.6 vi 在末行模式下的常用命令

命　令	作　用
:w	保存文件
:w !	强制保存文件
:q	退出 vi 编辑器
:q !	强制退出 vi 编辑器
:wq	保存退出
:wq !	强制保存退出
:e !	将文件还原为最初的状态（打开时的状态）
:ZZ	若文件没有被修改则退出，若文件已被修改，则保存退出
:w 文件名	将文件另存为一个副本
:r 文件名	读入特定的文件进行编辑
:*n*1,*n*2w 文件名	将文件的第 *n*1 行到 *n*2 行的内容另存为一个文件
:! 命令	暂时离开 vi，进入 Shell 执行命令。执行完成后回到 vi
:set nu	设置行号
:set nonu	取消行号设置

4.1.5 文件的恢复与暂存盘

在 vi 编辑器中使用 u 可撤销上一步操作，使用“:e!”可将文件回复至最初状态。vi 之所以能实现这些功能，是因为 vi 使用了暂存机制。

所谓暂存机制，是指 vi 将被编辑的文件打开时的状态暂存为一个临时文件，这个临时文件为“.被编辑文件名.swp”，该文件与被编辑文件位于同一目录下。当对文件进行编辑时，每一步操作步骤都会被记录，这样，vi 根据原始文件和操作步骤就可以将文件恢复到打开时的状态。

文件的临时文件只有在文件被编辑或 vi 非正常退出时产生。

4.2 vim 的附加功能

4.1 节中所描述的是 vi 与 vim 共有的功能，这些功能在日常管理工作中是很实用的。vim 是 vi 的升级版本，在 vi 功能的基础上新增加了一系列功能，包括块选择功能、多文件编辑功能、多窗口功能和语法提示功能等。本节将介绍常用的 vim 的扩展功能。

4.2.1 vim 的块选择功能

vim 允许用户实现块选择。vim 是一种行式编辑器，当需要复制数据时只能实现行式复制，当需要复制特定区块的数据时，行式编辑器就显得力不从心了。块选择功能提供了对于特定区块选择复制的功能。

图 4.2 是 lftp.conf 文件，当仅需要选择其中一个区块的内容时，就需要用到块选择功能。

```
## some useful aname
alias dir ls
alias less more
alias zless zmore
alias bzless bzmore
alias reconnect "close; cache flush; cd ."

## make prompt look better
set prompt "lftp \S\? \u\@\h:\w> "
## some may prefer colors (contributed by Matthew <mworr
)
#set prompt "\[\e[1;30m\][\[\e[0;34m\]f\[\e[1m\]t\[\e[37
[34m\]\u\[\e[0;34m\]\@\[\e[1m\]\h\[\e[1;30m\]:\[\e[1;34r
e[0m\] "
## Uncomment the following two lines to make switch cls
## cls the default.
#alias ls command cls
#alias hostls command ls

## default protocol selection
#set default-protocol/ftp.*     ftp
#set default-protocol/www.*     http
#set default-protocol/localhost file
```

选择特定区块的内容

图 4.2 lftp.conf 文件的内容

如何实现块选择功能呢？按 Ctrl+V 组合键，这时 vi 将自动进入 visual block 模式，可以使用方向键来控制光标选择特定的数据区块，如图 4.3 所示。

```
## some useful aname
alias dir ls
alias less more
alias zless zmore
alias bzless bzmore
alias reconnect "close; cache flush; cd ."

## make prompt look better
set prompt "lftp \S\? \u\@\h:\w> "
## some may prefer colors (contributed by Matth
)
#set prompt "\[\e[1;30m\][\[\e[0;34m\]f\[\e[1m\
[34m\]\u\[\e[0;34m\]\@\[\e[1m\]\h\[\e[1;30m\]:\
e[0m\] "
## Uncomment the following two lines to make sw
## cls the default.
#alias ls command cls
#alias hostls command ls

## default protocol selection
#set default-protocol/ftp.*      ftp
#set default-protocol/www.*      http
#set default-protocol/localhost file

-- VISUAL BLOCK --
```

利用光标选择了特定的数据区块

图 4.3　用光标选择数据区块

在选定数据区块之后可以按 y 键复制选中的内容，按 d 键删除选中的内容，这就是块选择功能，见表 4.7。这里注意，块选择功能是利用 Ctrl+V 组合键来实现。

表 4.7　vim 的块选择功能键

命　　令	说　　明	命　　令	说　　明
v	字符选择	y	复制选中内容
V	行选择	d	删除选中内容
Ctrl+V	块选择		

4.2.2　多文件编辑

所谓多文件编辑，是指在一个 vim 编辑界面中同时打开多个文件进行编辑，这样有利于在不同的文件之间进行比较和内容的相互复制。

例如，当对文件 lftp.conf 进行编辑时需要引用/etc/passwd 文件中的几行内容。这时利用多文件编辑机制将会很方便地实现该操作。

在 vim 中使用“:n　文件名”可以打开另一个文件进行编辑。

```
## some useful aname
alias dir ls
alias less more
alias zless zmore
alias bzless bzmore
alias reconnect "close; cache flush; cd ."

## make prompt look better
set prompt "lftp \S\? \u\@\h:\w> "
## some may prefer colors (contributed by Matthew <mwormald@optusnet.com.au>
)
#set prompt "\[\e[1;30m\][\[\e[0;34m\]f\[\e[1m\]t\[\e[37m\]p\[\e[30m\]] \[\e
[34m\]\u\[\e[0;34m\]\@\[\e[1m\]\h\[\e[1;30m\]:\[\e[1;34m\]\w\[\e[1;30m\]>\[\
e[0m\] "
## Uncomment the following two lines to make switch cls and ls, making
## cls the default.
#alias ls command cls
```

当需要在此处引用/etc/passwd文件中的内容时，使用“：n”打开文件进行编辑即可

```
15 #alias hostls command ls
16
17 ## default protocol selection
18 #set default-protocol/ftp.*      ftp
19 #set default-protocol/www.*      http
20 #set default-protocol/localhost file
"lftp.conf" 89L, 3245C                                  2,1           顶端
```

```
17 ## default protocol selection
18 #set default-protocol/ftp.*      ftp
19 #set default-protocol/www.*      http
20 #set default-protocol/localhost file
:n /etc/passwd
```

打开文件

```
 1 root:x:0:0:root:/root:/bin/bash
 2 bin:x:1:1:bin:/bin:/sbin/nologin
 3 daemon:x:2:2:daemon:/sbin:/sbin/nologin
 4 adm:x:3:4:adm:/var/adm:/sbin/nologin
 5 lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
 6 sync:x:5:0:sync:/sbin:/bin/sync
 7 shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
 8 halt:x:7:0:halt:/sbin:/sbin/halt
 9 mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
10 news:x:9:13:news:/etc/news:
11 uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin
12 operator:x:11:0:operator:/root:/sbin/nologin
13 games:x:12:100:games:/usr/games:/sbin/nologin
14 gopher:x:13:30:gopher:/var/gopher:/sbin/nologin
15 ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
16 nobody:x:99:99:Nobody:/:/sbin/nologin
17 rpm:x:37:37::/var/lib/rpm:/sbin/nologin
18 dbus:x:81:81:System message bus:/:/sbin/nologin
19 avahi:x:70:70:Avahi daemon:/:/sbin/nologin
20 mailnull:x:47:47::/var/spool/mqueue:/sbin/nologin
21 smmsp:x:51:51::/var/spool/mqueue:/sbin/nologin
22 nscd:x:28:28:NSCD Daemon:/:/sbin/nologin
23 vcsa:x:69:69:virtual console memory owner:/dev:/sbin/nologin
复制了 5 行                                             1,1           顶端
```

利用“5yy”复制5行内容

当复制了/etc/passwd 文件的 1～5 行内容之后，继续使用“:n lftp.conf”命令打开前一个文件，并将内容粘贴到文件中。

```
22 nscd:x:28:28:NSCD Daemon:/:/sbin/nologin
23 vcsa:x:69:69:virtual console memory owner:/dev:/sbin/nologin
:n lftp.conf
```

```
 1 ## some useful aname
 2 alias dir ls
 3 alias less more
 4 alias zless zmore
 5 alias bzless bzmore
 6 alias reconnect "close; cache flush; cd ."
 7 root:x:0:0:root:/root:/bin/bash
 8 bin:x:1:1:bin:/bin:/sbin/nologin
 9 daemon:x:2:2:daemon:/sbin:/sbin/nologin
10 adm:x:3:4:adm:/var/adm:/sbin/nologin
11 lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
12
13 ## make prompt look better
14 set prompt "lftp \S\? \u\@\h:\w> "
15 ## some may prefer colors (contributed by Matthew <mwormald@optusnet.com.au>
   )
16 #set prompt "\[\e[1;30m\][\[\e[0;34m\]f\[\e[1m\]t\[\e[37m\]p\[\e[30m\]] \[\e
   [34m\]\u\[\e[0;34m\]\@\[\e[1m\]\h\[\e[1;30m\]:\[\e[1;34m\]\w\[\e[1;30m\]>\[\
   e[0m\] "
17 ## Uncomment the following two lines to make switch cls and ls, making
18 ## cls the default.
19 #alias ls command cls
20 #alias hostls command ls
多了 5 行                                               7,1           顶端
```

使用p命令将复制的内容粘贴到此处

通过上面的例子可见，多文件编辑功能在对多个文件进行操作时是很方便的。该功

能的相关按键见表 4.8。

表 4.8　多文件编辑功能键

按　　键	含　　义
:n　文件名	编辑下一个文件
:N　文件名	编辑上一个文件
:files	列出当前 vim 打开的所有文件

```
alias zless zmore
alias bzless bzmore
alias reconnect "close; cache flush; cd ."
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin

## make prompt look better
set prompt "lftp \S\? \u\@\h:\w> "
## some may prefer colors (contributed by Matthew <mwormald@optusnet.com.au>)
#set prompt "\[\e[1;30m\][\[\e[0;34m\]f\[\e[1m\]t\[\e[37m\]p\[\e[30m\]] \[\e[
34m\]\u\[\e[0;34m\]\@\[\e[1m\]\h\[\e[1;30m\]:\[\e[1;34m\]\w\[\e[1;30m\]>\[\e[
0m\] "
## Uncomment the following two lines to make switch cls and ls, making
## cls the default.
#alias ls command cls
#alias hostls command ls

:files
  1 %a + "lftp.conf"                    第 7 行
  2 #    "/etc/passwd"                  第 1 行
请按 ENTER 或其它命令继续
```

显示当前vim所打开的文件

4.2.3　多窗口功能

多窗口功能是在同一个 vi 界面打开多个文件，每个文件占用一部分编辑窗口的应用。



```
# grub.conf generated by anaconda
#
# Note that you do not have to rerun grub after making changes to

# NOTICE:  You do not have a /boot partition.  This means that
#          all kernel and initrd paths are relative to /, eg.
#          root (hd0,0)
/boot/grub/grub.conf                                  1,1         顶端
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
/etc/passwd                                           1,1         顶端
alias bzless bzmore
alias reconnect "close; cache flush; cd ."

## make prompt look better
set prompt "lftp \S\? \u\@\h:\w> "
## some may prefer colors (contributed by Matthew <mwormald@optusnet.com.au>)
#set prompt "\[\e[1;30m\][\[\e[0;34m\]f\[\e[1m\]t\[\e[37m\]p\[\e[30m\]] \[\e[34m\]\u\[\e[0;3
4m\]\@\[\e[1m\]\h\[\e[1;30m\]:\[\e[1;34m\]\w\[\e[1;30m\]>\[\e[0m\] "
## Uncomment the following two lines to make switch cls and ls, making
lftp.conf                                             7,0-1       4%
```


使用多窗口可以很方便地实现文件不同部分内容的对比和引用，也可以实现不同文

件之间内容的对比和引用。

多窗口的启动和控制方法如下。

（1）启动多窗口编辑同一个文件，使用“:sp”在同一界面启动两个窗口，显示同一个文件。

```
23 #set ftp:sync-mode off
24
25 ## synchronous mode for broken servers and/or routers
:sp
```

将在这个界面启动两个窗口，显示同一个文件

```
 5 alias bzless bzmore
 6 alias reconnect "close; cache flush; cd ."
 7
 8 ## make prompt look better
 9 set prompt "lftp \S\? \u\@\h:\w> "
10 ## some may prefer colors (contributed by Matthew <mwormald@optusnet.com.au>)
11 #set prompt "\[\e[1;30m\][\[\e[0;34m\]f\[\e[1m\]t\[\e[37m\]p\[\e[30m\]] \[\e[34m\]\u\[\e[0;3
   4m\]\@\[\e[1m\]\h\[\e[1;30m\]:\[\e[1;34m\]\w\[\e[1;30m\]>\[\e[0m\] "
12 ## Uncomment the following two lines to make switch cls and ls, making
13 ## cls the default.
14 #alias ls command cls
15 #alias hostls command ls
lftp.conf                                                        7,0-1          5%
 5 alias bzless bzmore
 6 alias reconnect "close; cache flush; cd ."
 7
 8 ## make prompt look better
 9 set prompt "lftp \S\? \u\@\h:\w> "
10 ## some may prefer colors (contributed by Matthew <mwormald@optusnet.com.au>)
11 #set prompt "\[\e[1;30m\][\[\e[0;34m\]f\[\e[1m\]t\[\e[37m\]p\[\e[30m\]] \[\e[34m\]\u\[\e[0;3
   4m\]\@\[\e[1m\]\h\[\e[1;30m\]:\[\e[1;34m\]\w\[\e[1;30m\]>\[\e[0m\] "
12 ## Uncomment the following two lines to make switch cls and ls, making
13 ## cls the default.
14 #alias ls command cls
15 #alias hostls command ls
lftp.conf                                                        7,0-1          5%
:sp
```

窗口1

窗口2

（2）启动多窗口编辑不同的文件，使用“:sp　文件名”在同一个界面启动两个窗口，显示不同的文件。

```
24
25 ## synchronous mode for broken servers and/or routers
:sp /etc/passwd
```

```
 1 root:x:0:0:root:/root:/bin/bash
 2 bin:x:1:1:bin:/bin:/sbin/nologin
 3 daemon:x:2:2:daemon:/sbin:/sbin/nologin
 4 adm:x:3:4:adm:/var/adm:/sbin/nologin
 5 lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
 6 sync:x:5:0:sync:/sbin:/bin/sync
 7 shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
 8 halt:x:7:0:halt:/sbin:/sbin/halt
 9 mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
10 news:x:9:13:news:/etc/news:
11 uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin
12 operator:x:11:0:operator:/root:/sbin/nologin
/etc/passwd                                                      1,1            顶端
 5 alias bzless bzmore
 6 alias reconnect "close; cache flush; cd ."
 7
 8 ## make prompt look better
 9 set prompt "lftp \S\? \u\@\h:\w> "
10 ## some may prefer colors (contributed by Matthew <mwormald@optusnet.com.au>)
11 #set prompt "\[\e[1;30m\][\[\e[0;34m\]f\[\e[1m\]t\[\e[37m\]p\[\e[30m\]] \[\e[34m\]\u\[\e[0;3
   4m\]\@\[\e[1m\]\h\[\e[1;30m\]:\[\e[1;34m\]\w\[\e[1;30m\]>\[\e[0m\] "
12 ## Uncomment the following two lines to make switch cls and ls, making
13 ## cls the default.
14 #alias ls command cls
15 #alias hostls command ls
lftp.conf                                                        7,0-1          5%
"/etc/passwd" 34L, 1525C
```

（3）在不同窗口间切换可按以下组合键。

Ctrl+W+J：进入下面的窗口。

Ctrl+W+K：进入上面的窗口。

4.2.4 vim 的环境设置

在使用 vim 编辑文件时，vim 会主动将本次操作记录在~/.viminfo 文件中，以便下次打开文件时可以回到上次操作的终点。

同时，vim 每次启动时均会查找程序的初始化文件/etc/vimrc 和~/.vimrc，并根据初始化文件的设置来配置 vim 编辑器的属性。在这两个配置文件中/etc/vimrc 是对 vim 属性的定义，而~/.vimrc 配置文件是依每个用户不同的要求而定义的个性化的 vim 设置。表 4.9 中列出了一些常用的 vim 设置值，读者可以有选择地进行使用。

表 4.9 vim 的常用设置值

参　数	说　明
:set nu	设置行号。取消行号使用“:set　nonu”
:set hlsearch	设置选中值使用反白显示。不使用反白显示使用“:set　nohlsearch”
:set autoindent	设置自动缩排。不使用自动缩排使用“:set　noautoindent”
:set backup	设置自动备份。不使用自动备份使用“:set　nobackup”
:set ruler	设置在屏幕右下角显示状态行说明
:set showmode	设置在屏幕左下角显示操作说明，如“--INSTER--”
:set backspace=(012)	设置在编辑模式中是否可以使用 Backspace（退格键），2 表示可以使用退格键删除任意字符，0 和 1 表示可以使用退格键删除刚刚插入的字符
:syntax(off|on)	设置是否启用语法提示

本节介绍了 vi 编辑器的使用方法，掌握这部分内容需要不断地练习。vi 编辑器在日常管理工作中的地位很重要，其中命令模式下的查找替换操作、末行模式下的命令、多窗口操作以及配置文件设置是本章的难点。这里所谓难点是相对于其他操作而言的，特别是现在普遍采用 101 键盘，光标的移动往往使用方向键即可完成，所以对于需要记忆的快捷键和命令要做重点掌握。

第 2 篇　Linux 操作系统的基本管理

在熟悉了 Linux 操作系统的基本使用后，要将学习重点转移到操作系统的管理操作上了。操作系统管理是操作系统课程的基础课程，内容包括用户账户管理、文件系统管理、磁盘管理和系统资源管理等一系列主题，尽管这些主题所包含的内容很多，很繁杂，但是熟练地掌握这部分内容是成为合格的系统管理员的必备条件。

第5章 用户账号和组管理

Linux 是一个多用户操作系统，系统通过账号来标识用户，同时也是通过账号来控制用户对系统资源的访问权限。本章将就 Linux 系统中的账号管理问题展开讨论，通过对用户账号、组账号和登录信息查询等问题的讨论，使读者对账号管理工作有一个清晰的思路。

5.1 账号的基本知识

Linux 是一个多用户（Multi-User）的系统。用户就是指登录系统并运用系统资源工作的人，但并不一定负责系统的管理工作。管理员是一个特殊的用户，负责系统管理，所以用户在系统中的权限是被管理员规范的。用户可以通过网络登录，也可以通过主机所提供的虚拟终端登录系统。所以管理员的用户管理工作包括决定哪些用户能够登录系统，并给每一个用户分配一个登录系统的账号，设置操作权限等相关的管理。

一般情况下，一个用户在系统中应该只有一个账号，系统把不同账号当做是不同的用户，虽然这些账号可能属于同一个人。本书中提到的用户对系统而言就是一个账号，所以下面提到的用户和账号的含义是相同的。

若系统不连接网络，并且只有一个用户，也就是单一用户使用时，在不同工作环境下用户也有必要以不同账号登录系统。最简单的情况就是：用户必须以 root 账号登录系统以取得最大权限，才能修改系统及进行相关设置，而日常应用则应该以普通用户的账号登录，以免误操作对系统造成无法修复的损害。

5.2 用 户 账 号

账号管理包括账号数据文件管理，添加账号、密码和用户信息，以及在必要时删除账号等操作。

5.2.1 管理用户账号数据文件

到目前为止我们一直在使用 root 这个账号登录系统进行操作，这主要是从学习系统功能的角度出发，使用 root 账号可以避免因权限问题造成的干扰，有利于读者理清思路。但是，root 账号是一个拥有所有权限的账号，使用该账号登录系统如果发生误操作则很

有可能破坏整个系统。因此，在实际工作环境中，除非是需要对系统作更改，平时的工作应该以普通用户账号登录系统。那么普通用户账号该如何添加呢？接下来开始介绍账号的添加操作。

首先，用 useradd 命令添加用户账号，然后再用 passwd 命令设置账号密码，经过这两个步骤，就能成功地设置一个新账号，并且可以成功地登录系统。下面先简单介绍这两个命令的用法。

useradd 命令用于添加用户账号，其语法结构为：

useradd 账号名称

passwd 命令用于设置特定用户账号的密码，其语法结构为：

passwd 账号名称

【示例】为当前系统添加 user1 账号并设置密码。

```
[root@hero ~]# useradd user1
[root@hero ~]# passwd user1
Changing password for user user1.
New UNIX password:     ←要求用户输入密码，密码不回显
BAD PASSWORD: it is too simplistic/systematic
Retype new UNIX password:  ←要求用户再次输入密码，密码不回显
passwd: all authentication tokens updated successfully.
```

注意，当为用户账号设置密码时，出于安全性考虑，Linux 不会在屏幕上回显所输入的密码，但用户输入的密码却已被操作系统所接受。同时，两次输入的密码必须一致，否则会因确认密码错误而导致用户账号创建失败。另外，一个用户账号如果未设置密码，默认的情况下是不允许登录系统的。

设置完用户账号和密码后可以使用 login 命令来登录系统。

```
CentOS release 5.5  ( Final )
Kernel 2.6.18-194.el5 on an i686
hero login: user1
Password:
[user1@hero ~]$
```

这里应注意，普通用户账号的命令提示符为“$”。

以上是创建用户的基本操作。当创建完用户之后，用户信息会保存在/etc/passwd 文件中，用户可以使用 cat 命令显示文件的内容或以编辑工具查看和修改文件的内容。以下是该文件的内容：

```
[root@hero ~]# cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
…
user1:x:510:512::/home/user1:/bin/bash
```

/etc/passwd 文件是 Linux 系统中非常重要的账号信息数据库文件，文件中的每一行都代表一个账号的数据，用户可以看到文件中有 root 以及刚刚添加的 user1 等用户账号，此外还有系统在安装时自动创建的标准用户（Standard Users），如 bin、daemon 等，而标准用户账号并不是日常使用的账号，故暂时不在这里讨论。

在创建完用户账号后，下面重点讨论用户账号数据文件/etc/passwd。这个文件在账号管理工作中是非常重要的，用户账号的相关信息都是由这个文件定义的。

在/etc/passwd 文件中每一个账号都由 7 个字段的数据组成，以“:”分隔，其格式如下：

账号名称：密码：UID：GID：用户信息：主目录：登录 Shell

如，root 账号信息为：

```
root :     x    :  0  :    0  : root  :   /root  :  /bin/bash
  ↑        ↑       ↑         ↑     ↑           ↑             ↑
 账号名称   密码    UID      GID  用户信息      主目录       登录 Shell
```

对上述各字段说明如下：

（1）账号名称：登录系统时使用的名称。

（2）密码：登录密码。这个字段显示的是加密后的乱码而不是真实的密码，这可以避免用户账户密码泄露给不该知道的人。用户在这个字段常常看到的是 x。

```
root : x :0:0:root:/root:/bin/bash
bin : x :1:1:bin:/bin:/sbin/nologin
daemon : x :2:2:daemon:/sbin:/sbin/nologin
adm : x :3:4:adm:/var/adm:/sbin/nologin
lp : x :4:7:lp:/var/spool/lpd:/sbin/nologin
sync : x :5:0:sync:/sbin:/bin/sync
shutdown : x :6:0:shutdown:/sbin:/sbin/shutdown
halt : x :7:0:halt:/sbin:/sbin/halt
mail : x :8:12:mail:/var/spool/mail:/sbin/nologin
news : x :9:13:news:/etc/news:
uucp : x :10:14:uucp:/var/spool/uucp:/sbin/nologin
```

这表示密码经过 shadow passwords 保护以增加安全性。可以使用 pwconv 命令执行保护或用 pwunconv 命令停止保护。在 shadow passwords 的保护下，所有有关密码的数据及设置都存放在/etc/shadow 文件内，在 passwd 文件中就只能看到 x。

（3）UID（用户标识符，User ID）：是系统用于识别用户账号的标识，所以每一个账号都会有一个 UID，且 UID 具有唯一性，不会与其他账号的 UID 重复。root 的 UID 为 0，1~99 是系统的标准账号，普通用户可以自由运用从 100 开始的 UID。CentOS 系统默认的普通用户可以自由使用的 UID 是 500 以上，当用户使用 useradd 命令添加一个账号时，会自动从 500 开始给定 UID。自定义用户的 UID 范围为 500~65 535，当前的 Linux 内核（2.6.x 版）已经可以支持 4 294 967 295（$2^{32}-1$）这么大的 UID 号码了。对于 100~499 这

一 UID 范围系统往往会保留给一些服务使用。

（4）GID（组标识符，Group ID）：每个用户账号都属于一个原始组（Primary Group），同组的用户 GID 也会相同。CentOS 使用专门用户组（UPG，User Private Group）配置法，使用 useradd 指令添加账号时，系统默认会添加一个专门用户组作为该用户的原始组，同时以账号名称作为组名称，该账号就是这个原始组的唯一成员。

（5）用户信息：该字段中可以记录关于用户的姓名、电话等信息。这个字段通常没有记录，详细配置在 5.2.3 节中介绍。

（6）主目录：定义用户的主目录，通常是/home/name，name 与用户账号名称相同。

（7）登录 Shell：定义用户登录系统后激活的 Shell，默认是 bash。需要注意的是，当将/sbin/nologin 作为默认的登录 Shell 时，可以使账号无法登录系统。

上面提到了 shadow passwords 机制，该机制很好地保证了用户口令不会因/etc/passwd 文件的所有用户可读取的特点而导致泄露。shadow passwords 机制会将用户口令转存到/etc/shadow 文件中，/etc/shadow 文件根据/etc/passwd 文件生成，只有超级用户才能查看其内容。在/etc/shdadow 文件中，口令使用 MD5 算法加密。MD5 算法是一种单向加密算法，破解难度很大。这就很好地保证了账户密码的安全性。同时/etc/shdadow 文件中添加了很多密码的限制性参数。

```
[root@hero ~]# cat /etc/shadow
root:$1$IltuxymH$uIds/Z7SIQDUypwCtuzVI.:15047:0:99999:7:::
…
user1:$1$24Io1nBq$oeCFMqlQDuaEQgRyNuDDK1:15062:0:99999:7:::
```

5.2.2 添加用户账号与设置密码

1．创建新账号

添加用户账号的指令为 useradd，有些版本的 Linux 使用 adduser 命令，在 CentOS 中 adduser 命令作为 useradd 命令的符号链接而存在，所以使用 useradd 与 adduser 是相同的。

useradd 命令只有 root 账号有权使用，它会导致在系统中创建一个新的账号。账号的创建过程会进行下列动作。

（1）在/etc/passwd 文件中添加新账号的相关数据。

（2）在/etc/group 文件中添加新用户的原始组。

（3）创建用户主目录，默认目录为/home/name，name 与用户账号名称相同。

（4）将/etc/skel 目录中以“.”开头的用户骨架文件复制到用户主目录，这些文件是一些环境设置的文件。

（5）如果系统启用 shadow passwords 保护，则在/etc/shadow 文件中添加账号密码的配置信息。

下面对 useradd 命令的使用加以说明。

【语法】**useradd [-选项] 用户名**

选项说明如下：

-d dir：设置用户主目录。dir 为用户要设置的主目录名称，主目录名称默认是账号名称。

-e date：设置此账号的有效日期到 date，date 的格式为 MM/DD/YYYY，例如，08/01/2011 表示到 2011 年 8 月 1 日这个账号就无法再登录系统。这项功能必须在系统使用 shadow passwords 功能时才能使用。

-f days：在密码终止的 days 天数后停止这个账号。该功能也必须要由 shadow passwords 功能执行。

-g group：设置该账号所属的原始组。可以使用 group 名称或 GID 设置。

-G group：设置该账号所属的附属组，即该账号为所属组的成员，可以使用 group 名称或 GID 设置，组之间需要使用“,”分隔。

-m：若用户主目录不存在，则一并创建用户主目录。

-k dir：在使用-m 选项的同时，将 dir 中的文件复制到用户主目录下，若不设置 dir 参数时，则系统默认将复制/etc/skel 目录下的文件。这项功能必须在-m 同时被执行时才有效。

-s Shell：设置登录启动后的 Shell。

-u UID：设置此账号的 UID。

【示例】以 useradd 命令创建一个新账号 user2。

```
# useradd  user2
```

创建完 user2 账号后，会在/etc/passwd 文件中得到 user2 的账户信息。

```
user2:x:501:501::/home/user2:/bin/bash
```

系统自动设置 user2 的 UID 为 501，并创建 user2 的专门用户组（UPG，User Private Group）为原始组，GID 为 501，并设置用户主目录为/home/user2。注意，由于当前系统启用 shadow passwords 密码保护机制，所以此处无法显示密码状态，可以使用 pwunconv 关闭 shadow passwords 机制，这时，user2 用户的密码字段为“!!”，表示密码未设置。

```
user2:!!:501:501::/home/user2:/bin/bash
```

【示例】添加一个账号 user3，并定义 user3 的所属组为 user2。

```
# useradd  -g  user2  user3
```

或

```
# useradd  -g  501   user3
```

```
user2:!!:501:501::/home/user2:/bin/bash
user3:!!:502:501::/home/user3:/bin/bash
```

2. 设置密码

添加用户账号后还需要设置用户密码，否则无法登录系统。第一次设置用户密码需要由 root 账号来完成，应使用 passwd 命令设置用户的密码。

下面对 passwd 命令的使用加以说明。

【语法】**passwd　[选项]　[用户名]**

选项说明如下：

-l：锁定密码，用户将无法登录。

-u：将由-l 选项锁定的密码解锁，用户可以用原密码登录。

-d：删除密码，使该用户为密码未设置或空白密码状态。

-S：显示该用户账号的密码状态。

用户名：是要修改或设置密码的账号名。只有 root 可以设置其他用户账号的密码。普通用户只能修改自己的账号密码。当修改自己的账号密码时，在 passwd 命令中就可以不使用用户名参数了。

【示例】设置 user2 账号的密码。

```
[root@hero ~]# passwd user2
Changing password for user user2.
New UNIX password:
BAD PASSWORD: it is too simplistic/systematic
Retype new UNIX password:
passwd: all authentication tokens updated successfully.
```

【示例】删除 user2 账号的密码。

```
[root@hero ~]# passwd -S user2
user2 PS 2011-12-30 0 99999 7 -1 (Password set, MD5 crypt.) ←密码已设置
[root@hero ~]# passwd -d user2
Removing password for user user2.
passwd: Success
[root@hero ~]# passwd -S user2
user2 NP 2011-12-30 0 99999 7 -1 (Empty password.) ←密码为空
```

5.2.3　查看及修改用户信息

1. 查看用户信息

使用 finger 命令可以查看用户的相关信息，包括用户主目录、Shell、登录数据以及在 passwd 文件的账号数据记录中的用户信息字段，包括姓名和电话等。

下面对 finger 命令的用法加以说明。

【语法】**finger [选项]　用户名 1　…用户名 *n***

选项说明如下：

-l：以长格式显示数据，系统默认就是以长格式显示数据。

-m：关闭根据用户名查找账号的功能，在不加-m 的情况下，finger 可以以 Name 字段的姓名指定用户。

-p：不显示 plan（计划）与 project（项目）描述。

-s：以短格式显示数据。

用户名列表：finger 命令可以同时显示多个用户的用户信息。如果不添加用户名参数，将显示当前用户的用户信息。

【示例】查看 user2 的用户信息。

```
[root@hero ~]# finger user2
Login: user2                              Name: (null)
Directory: /home/user2                    Shell: /bin/bash
Never logged in.
No mail.
No Plan.
```

由输出可见，user2 账号的用户信息很少。出现这种情况的原因是还未对 user2 账号设置用户信息，所有显示的用户信息均是来源于系统默认。那么，应该如何设置用户信息呢？可以通过修改/etc/passwd 文件中的“用户信息”字段来修改用户信息，也可以使用 chfn 命令来修改用户信息。

2. 修改用户信息

用户信息提供了用户的联系方式、计划工作等相关信息，这些信息为管理员管理多用户系统提供了很好的信息参照。默认的情况下系统仅记录用户账号的一些基本信息。为了能更好地描述用户，可以使用 chfn 指令来设置和修改用户信息。

chfn 命令用于设置和修改用户信息，其语法结构为：

【语法】 **chfn ［用户名］**

这里用户名是一个可选项，当不使用用户名参数时将设置或修改当前用户的用户信息。

【示例】设置 user2 用户的用户信息，并查看该用户信息。

```
[root@hero ~]# chfn user2
Changing finger information for user2.
Name []: tom                    ←添加姓名
Office []: 5001                 ←添加用户办公室编号
Office Phone []: 8365974        ←添加用户办公室电话
Home Phone []: 5568754          ←添加用户家庭联系电话

Finger information changed.
[root@hero ~]# finger user2
Login: user2                              Name: tom
Directory: /home/user2                    Shell: /bin/bash
```

```
Office: 5001, 836-5974                    Home Phone: 556-8754
Never logged in.
No mail.
No Plan.
```

在用户信息中还存在 plan 和 project 两项内容，这两项的内容记录在用户主目录下的.plan 和.project 文件中，而 Name、Office 和 Home Phone 的数据是 passwd 文件的账号数据记录中用户信息字段的内容，三项数据间以“,”分隔。

```
user2:x:501:501:tom,5001,8365974,5568754:/home/user2:/bin/bash
```

【示例】修改 user2 的.plan 和.project 文件，增加该账户工作计划和项目说明。

```
# vi  ~user2/.plan                        添加该账户的工作计划
#vi  ~user2/.project                      添加该账户参加的项目情况
[root@hero ~]# finger user2               查看 user2 的用户信息
Login: user2                              Name: tom
Directory: /home/user2                    Shell: /bin/bash
Office: 5001, 836-5974                    Home Phone: 556-8754
Never logged in.
No mail.
Project:
Business inspection
Database migration in preparation
Plan:
Network flow monitoring
File system monitoring
Memory monitoring
Load monitoringn
```

5.2.4 修改用户账号的相关设置

root 有权修改用户账号的设置，除直接编辑/etc/passwd 文件外，还可以使用 usermod 命令来进行修改。usermod 命令的作用就是修改账户设置。

下面对 usermod 命令的使用加以说明。

【语法】#usermod ［选项］ 账号名称

选项说明如下：

-d dir：设置用户主目录。dir 为用户要设置的主目录名称，主目录名称默认是账号名称。

-e date：设置此账号的有效日期到 date，date 的格式为 MM/DD/YYYY，例如 08/01/2011 表示到 2011 年 8 月 1 日这个账号就无法再登录系统。这项功能必须在系统使用 shadow passwords 功能时才能使用。

-f days：在密码终止的 days 天数后停止这个账号。该功能也必须要由 shadow passwords 功能执行。

-g group：设置该账号所属的原始组。可以使用 group 名称或 GID 设置。

-G group：设置该账号所属的附属组，即该账号为所属组的成员，可以使用 group 名称或 GID 设置，组之间需要使用“，”分隔。

-m：若用户主目录不存在，则一并创建用户主目录。

-k dir：在使用-m 选项的同时，将 dir 中的文件复制到用户主目录下。若不设置 dir 参数，则系统默认将复制/etc/skel 目录下的文件。这项功能必须在-m 同时被执行时才有效。

-s Shell：设置登录启动后的 Shell。

-u UID：设置此账号的 UID。

-i name：更改账号名称，必须在该账号未登录系统时才能使用。

账号名称：必须使用账号名称参数。

5.2.5 用户账号停用

停用用户账号分为以下几种情况。

（1）暂时停止登录系统的能力，日后再恢复。

（2）从系统中删除账号，但保留用户的文件。

（3）完全删除账号及该用户的所有文件。

1. 暂停账号

要暂时停止用户登录系统的能力，只需要利用编辑工具将 passwd 文件中的密码字段加上“*”，该账号就不能登录。若是在 shadow passwords 保护下，就需要在/etc/shadow 文件中的第二个字段上添加“*”才起作用。只要把“*”删除，账号即可恢复登录能力。

另一个比较简单的方法就是用 passwd 命令将该账号密码锁定起来。passwd 命令在 5.2.1 节和 5.2.2 节已经介绍过了。

【示例】暂停 user2 账号的登录能力。

```
[root@hero ~]# passwd -l user2
Locking password for user user2.
passwd: Success
```

当需要恢复账号的登录能力时可以使用“passwd －u 账号名称”来解锁用户密码。

【示例】恢复 user2 账号的登录能力。

```
[root@hero ~]# passwd -u user2
Unlocking password for user user2.
passwd: Unsafe operation (use -f to force).
```

2. 删除账号

要删除一个用户账号，可以直接将 passwd 文件中该用户的记录整行删除，或使用 userdel 命令。一般使用 userdel 命令效果会比较好。

删除用户账号后，系统内该用户的文件可以用 chown 指令改变文件的所有权，以及移动文件的位置让别的用户接替这些文件。

下面对 userdel 命令的用法加以说明。

【语法】 **userdel [-r] 账号名称**

选项说明如下：

-r：删除用户账号时一并将该用户的主目录以及主目录内的所有文件删除。

账号名称：要删除的账号名称。

【示例】删除 user2 账号，但保留其文件。

```
# userdel  user2
```

3. 完全删除账号

若是需要连同用户的文件一并删除，可以使用以下命令：

```
userdel  -r  账户名称
```

该命令除删除主目录的文件外，如果该用户还使用了 E-mail，则在/var/spool/mail/目录下的用户邮件也会被一并删除。

5.3 组

运用组可以方便管理用户和工作。例如，在一个机构中，可以根据部门来分组，同一部门的用户在系统中就是同一组；或者在一项工作计划中，将参与的人员编为同一组的成员，计划相关的文件就可以开放给这个组，每个成员都有进入文件的相同权限。

5.3.1 管理组数据的文件

有关组的数据存储于/etc/group 文件中，利用 vim 等编辑工具可以修改文件的内容。group 文件的内容如下：

```
[root@hero ~]# cat /etc/group
root:x:0:root
bin:x:1:root,bin,daemon
…
user1:x:512:
```

group 文件与 passwd 文件的格式相似，group 文件的每一行记录了一个组的数据，如上面的 root 组、用户 user1 组等，也有系统创建的标准组（Standard Groups），如 bin 组和 deamon 组等。

每个组的数据包括 4 个字段，字段间也是以“：”分隔，内容说明如下：

组名称：组密码：GID：组成员

（1）组名称：该组的名称，如 root、user1。

（2）组密码：设置加入组的密码，大多数情况下不使用组密码，所以这个字段通常没有用。如果使用了组密码且系统已启用 shadow passwd 机制时，组密码将记录在/etc/gshadow 文件中。

（3）GID：组标识符（group id），与用户的 UID 一样，每个组也有自己的 GID 供系统识别，且不同组的 GID 不同。

（4）组成员：这一字段记录了属于该组的成员，大部分用户的专门组字段是空的。用户可以根据需求加入某个组，成为该组的成员。

5.3.2 添加、删除组与修改组数据

在添加用户时，系统默认会创建用户的专门组，而用户可以根据需要自行添加或删除组。要添加或删除组，可以直接编辑/etc/group 文件或使用 groupadd 与 groupdel 命令。

1. 添加组

groupadd 命令用于在系统内创建新组。

下面对 groupadd 命令的使用加以说明。

【语法】**groupadd [选项] 组账号名称**

选项说明如下：

-g GID：更改组标识符为新的 GID。

-f：强制接受添加的组标识符为重复的标识符。

组账号名称：是一个必选的参数，用于定义待添加的组名称。

【示例】在系统内添加 xyproject 组，且 GID 为 600。

```
[root@hero ~]# groupadd -g 600 xyproject
[root@hero ~]# tail /etc/group
user6:x:505:
user7:x:506:
user8:x:507:
user9:x:508:
project:x:509:
lab1:x:510:
user:x:511:
user1:x:512:
usery:x:513:
xyproject:x:600:
```

2. 删除组

groupdel 用于删除组账号。

下面对 groupdel 命令的用法加以说明。

【语法】**groupdel 组账号名称**

【示例】删除系统中的 xyproject 用户组。

```
[root@hero ~]# tail /etc/group
user:x:511:
user1:x:512:
usery:x:513:
xyproject:x:600:
[root@hero ~]# groupdel xyproject
[root@hero ~]# tail /etc/group
user:x:511:
user1:x:512:
usery:x:513:
```

3. 修改组设置

groupmod 命令用于修改组设置，包括 GID 和组名称等信息。

下面对 groupmod 命令的用法加以说明。

【语法】**groupmod [选项] 组账号名称**

选项说明如下：

-g GID：更改 GID。

-n name：更改组账号名称。

-o：强制接受更改的 GID 为重复值。

组账号名称：是一个必选参数，用于定义待修改的组账号的名称。

【示例】修改 xyproject 组的名称为 softwarepro，且强制 GID 为 500。

```
[root@hero ~]# tail -n 1 /etc/group
xyproject:x:600:
[root@hero ~]# groupmod -n softwarepro -g 500 -o xyproject
[root@hero ~]# tail -n 1 /etc/group
softwarepro:x:500:
```

5.3.3 添加与删除组用户

可以利用组来组织用户。本节说明如何将用户加入组，以及如何从组中删除用户。

组用户的管理可以使用 gpasswd 命令。gpasswd 命令原本是用来设置组密码的命令，但是在专门用户组的模式下组密码并不常用。相对而言，gpasswd 命令用来管理组内用户的功能倒是很方便，它可以将用户加入到组中或从组中删除。该命令只有 root 或组管理员有权使用。

下面对 gpasswd 命令的用法加以说明。

【语法】**gpasswd [选项] [用户账号名称] [组账号名称]**

选项说明如下：

-a：将用户账号加入到组账号中。

-d：将用户账号从组账号中删除。

-r：取消组密码。

用户账号名称：表示要加入或删除的用户账号。

组账号名称：表示要加入或删除用户的组名称，或设置组密码的组名称。

【示例】将用户 user1 加入 softwapro 组内。

```
[root@hero ~]# tail -n 1 /etc/group
softwarepro:x:500:
[root@hero ~]# gpasswd -a user1 softwarepro
Adding user user1 to group softwarepro
[root@hero ~]# tail -n 1 /etc/group
softwarepro:x:500:user1
```

5.4 深入掌握用户与组操作

5.4.1 有效用户组与用户原始组

Linux 采用专门用户组配置方法，即为每个新建用户创建一个专属于该用户的组，默认的情况下 GID 与组名与 UID 和用户名相同。同时，在创建用户时也可以将用户加入多个组，或者使用 usermod 命令将用户添加至多个组。

当用户同时属于多个组时，该用户究竟将以哪个组的身份进行操作呢？这其实涉及两种情况。第一种情况是，当这种具有多组身份的用户对文件进行权限性操作时，如访问文件、修改文件或执行文件等，用户将以各组权限的集合对文件进行操作，如图 5.1 所示。

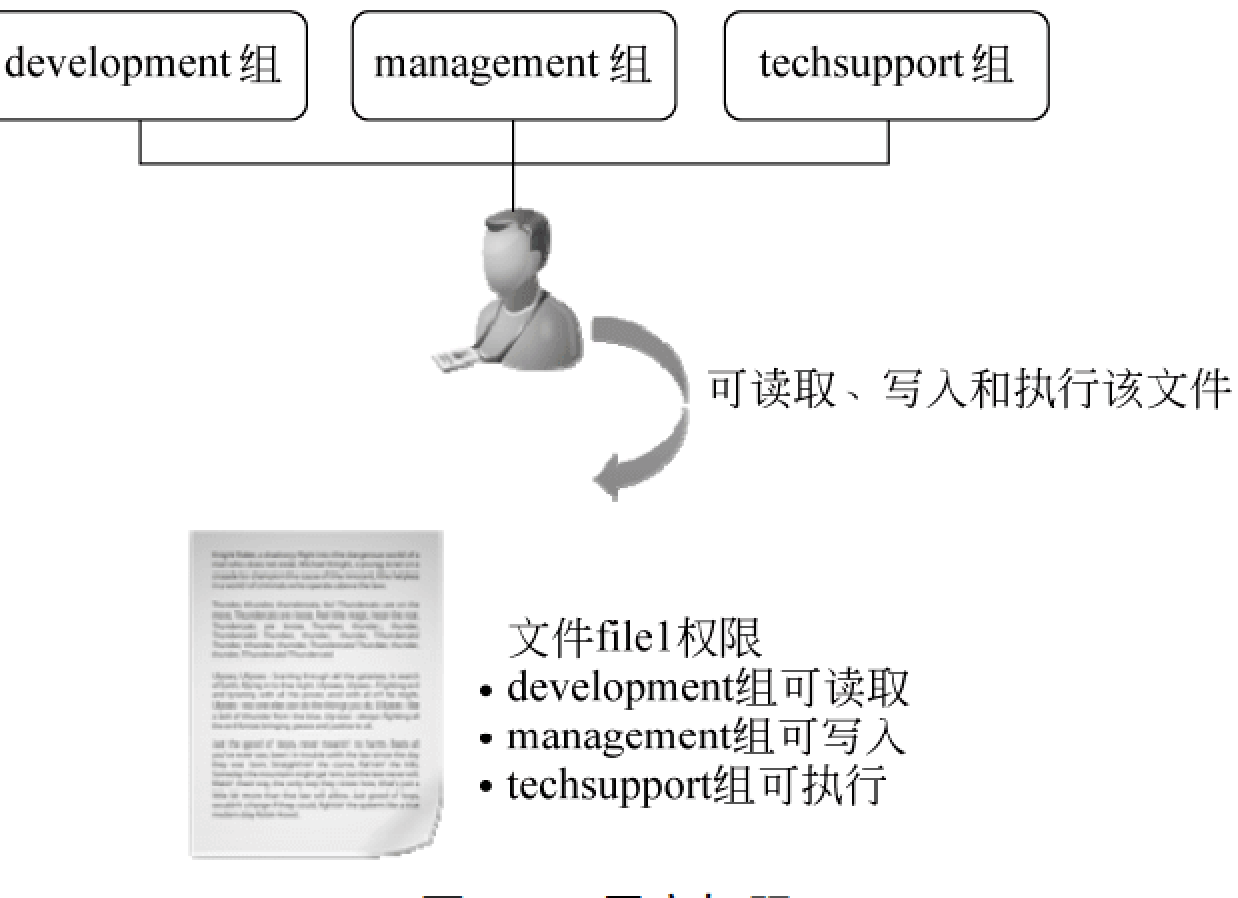


图 5.1　用户权限

在图 5.1 中，用户 user1 同时属于 development 组、management 组和 techsupport 组。当文件 file1 对 development 组有读取权限、对 management 组有写入权限、对 techsupport 组有执行权限时，由于用户 user1 同时属于这 3 个组，所以 user1 对这个文件具有这 3 个

组权限的集合，即 development 组的读取权限+management 组的写入权限+techsupport 组的执行权限。

另一种情况是，当这种具有多组身份的用户进行定义性操作时，如创建文件，用户将以有效用户组的身份进行操作。

如图 5.2 所示，因为用户 user1 的有效用户组是 development，所以当 user1 创建一个文件时，该文件的属组为 development。

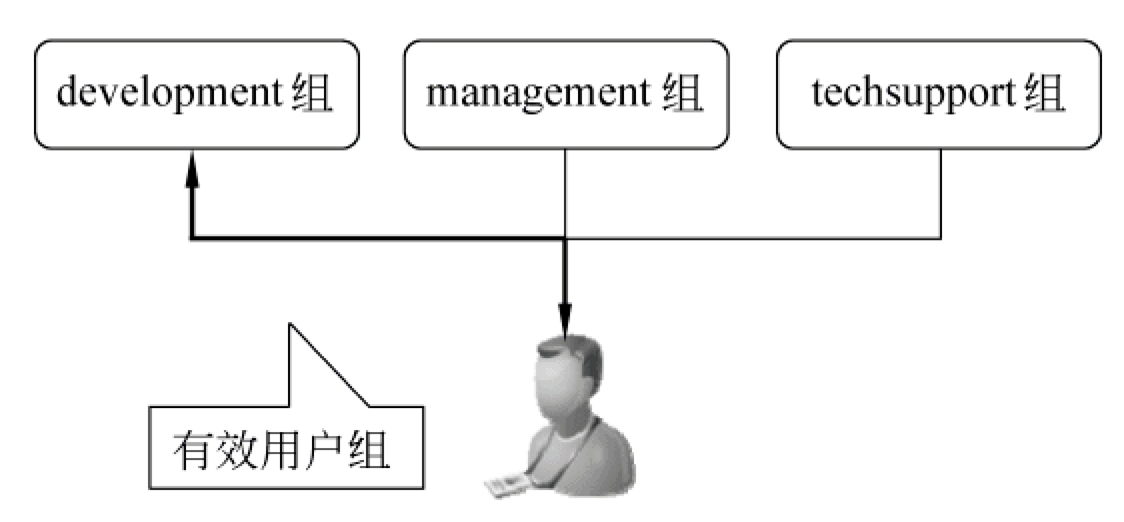


图 5.2　有效用户组

如何查看用户的属组分布情况呢？可以使用 groups 命令。

groups 命令用于查看当前用户的属组情况。

下面对 groups 命令的用法加以说明。

【语法】**groups**

【示例】查看当前用户 user1 的属组分布情况和有效用户组。

```
[user1@hero~]$ groups
development management techsupport
```

第一个出现的组 development 为有效用户组。

【示例】使用 user1 创建文件 file1，并查看文件的属主和属组。

```
[user1@hero~]$ touch file1
[user1@hero~]$ ll file1
-rw-r--r--1 user1 development 0 09-14 15：12 file1
```

注意，这时文件的属主是 user1，属组为 development，这是因为用户 user1 的有效用户组是 development。

当需要修改当前用户的有效用户组时，需要使用 newgrp 命令。

下面对 newgrp 命令的用法加以说明。

【语法】**newgrp　组名**

【示例】将 user1 用户的有效用户组定义为 techsupport。

```
[user1@hero~]$newgrp  techsupport
[user1@hero~]$groups
techsupport development management
```

注意，这时用户的有效用户组为 techsupport。

【示例】使用 user1 创建文件 file2，并查看文件的属主和属组。

```
[user1@hero~]$ touch  file2
[user1@hero~]$ ls -l  file2
-rw-r--r--1 user1 techsupport 0 09-14 15：20 file2
```

由于 user1 用户的有效用户组发生变化，这时在进行定义性操作时，将以新的有效用户组身份进行。

5.4.2　创建用户时的默认配置文件

在 5.2.2 节中介绍了如何创建用户，以及用户创建过程中 Linux 操作系统默认所进行的操作，如在 /home/目录下创建用户主目录，将/etc/skel 目录中的骨架文件复制到用户主目录，为用户分配 UID 和 GID 等。那么这些默认操作都是谁来决定的呢？

实际上 useradd 命令会参考一个配置文件/etc/default/useradd，并根据这个配置文件中的定义决定默认的用户主目录的位置以及用户骨架文件的位置等。

/etc/default/useradd 文件的内容为：

```
#useradd defaults file
GROUP=100                   默认的用户组
HOME=/home                  默认的用户主目录位于/home 目录下
INACTIVE=-1                 默认密码过期的宽限时效
EXPIRE=                     默认账号的过期日期
SHELL=/bin/bash             默认的用户登录 Shell
SKEL=/etc/skel              默认的用户骨架文件的存储位置
```

这里需要对“GROUP=100 ”默认的用户组进行一个说明。useradd 命令在建立用户时，对用户的原始组有着两种不同的处理机制。

一种机制以 Red Hat Linux 为代表。当新建用户时，若未指定原始用户组，则系统自动建立一个与用户名和 UID 相同的组账号。

另一种机制以 SuSE 为代表。新建用户时，默认不会建立新用户组，而是以/etc/default/useradd 文件中的 GROUP 命令定义的值作为用户的初始用户组。

5.4.3　UID/GID 的分配

在创建用户时是如何分配 UID 和 GID 的呢？useradd 命令会依据配置文件/etc/login.defs 的内容来决定 UID 与 GID 的分配。

/etc/login.defs 文件的内容如下：

```
[root@hero ~]# cat /etc/login.defs
MAIL_DIR         /var/spool/mail
PASS_MAX_DAYS    99999
PASS_MIN_DAYS    0
PASS_MIN_LEN     5
PASS_WARN_AGE    7
```

```
UID_MIN                    500
UID_MAX                  60000
GID_MIN                    500
GID_MAX                  60000
CREATE_HOME      yes
UMASK            077
USERGROUPS_ENAB yes
MD5_CRYPT_ENAB   yes
```

各行的意义如下：

MAIL_DIR /var/spool/mail：定义了默认用户邮件的存储位置。

PASS_MAX_DAYS 99999：定义了默认密码更新的天数为不需要更新。

PASS_MIN_DAYS 0：定义了默认密码不可更改天数为随时更改。

PASS_MIN_LEN 5：定义了默认密码的最小长度为 5 位。

PASS_WARN_AGE 7：定义了默认密码更改期限前的警告信息发出时间为提前 7 天。

UID_MIN 500：定义了用户最小的 UID 为 500。

UID_MAX 60000：定义了用户最大的 UID 为 60 000，可见系统可容纳 60 000 个用户账号。

GID_MIN 500：定义了组最小的 GID 为 500。

GID_MAX 60000：定义了组最大的 GID 为 60 000，可见系统可容纳 60 000 个组账号。

CREATE_HOME yes：定义是否自动建立用户主目录，yes 为自动建立。注意 Red Hat Linux 会自动建立用户主目录，而 SuSE Linux 即使此处为 yes 也不会自动建立用户主目录，而需要使用“-m”选项。

UMASK 077：umask 的含义和相关内容将在后续章节中介绍，这里先清楚该命令定义了文件或目录的默认权限即可。

USERGROUPS_ENAB：专门用户组配置是否开启，ENAB 表示开启，DISA 表示不开启。

注意，SuSE 的 UID_MIN 是从 1000 开始定义的。

当系统新增一个用户时，这个用户的 UID 的产生方式是，如果/etc/passwd 文件中的 UID 没有大于/etc/login.defs 文件中 UID_MIN 命令定义的值的，则使用 UID_MIN 所定义的值作为用户的 UID。如果/etc/passwd 文件中的 UID 存在有大于/etc/login.defs 文件中 UID_MIN 命令定义的值的，则使用 UID+1 作为新账号的 UID 值。

当需要建立一个用户账号，要求其 UID 存在小于 UIM_MIN 所定义的值时，可以使用“useradd -r 账号名称”命令。注意，这里的“-r”选项将忽略/etc/login.defs 文件中 UID_MIN 的定义，同时需要注意使用“-r”选项将不会建立用户主目录。如果需要同时建立用户主目录，则需要使用“-m”选项。

5.4.4 查看用户的 ID 信息

用户的 UID、GID 等信息可以使用 id 命令查询。

id 命令用于显示用户的 UID 与 GID 等相关信息。

下面对 id 命令的用法加以说明。

【语法】**id [用户名]**

id 命令单独使用时将显示当前用户的 UID、GID 等信息。当需要显示其他用户的 ID 信息时，则需要添加用户名参数。

【示例】显示当前用户的 id 信息。

```
[root@hero ~]# id
uid=0(root)gid=0(root)groups=0(root),1(bin),2(daemon),3(sys),4(adm),
6(disk),10(wheel)
```

【示例】显示 user1 用户的 id 信息。

```
[root@hero ~]# id user1
uid=510(user1) gid=512(user1) groups=512(user1),500(softwarepro)
```

5.4.5 设置用户密码策略

有关用户账户的密码策略可以使用 passwd 命令来进行定义，passwd 命令可以修改账号密码的不可更改天数、密码更新的天数以及密码更新前警告的天数。

【语法】**passwd [选项] 用户名**

选项说明如下：

-n 天数：定义密码的不可更改天数。

-x 天数：定义密码更新的天数。

-w 天数：定义密码更新前警告的天数。

【示例】设置 user1 账户 5 天内不允许修改密码。

```
[root@hero ~]# cat /etc/shadow | grep user1
user1:$1$24Io1nBq$oeCFMqlQDuaEQgRyNuDDK1:15062:0:99999:7:::
[root@hero ~]# passwd -n 5 user1
Adjusting aging data for user user1.
passwd: Success
[root@hero ~]# cat /etc/shadow | grep user1
user1:$1$24Io1nBq$oeCFMqlQDuaEQgRyNuDDK1:15062:5:99999:7:::
```

【示例】设置 user1 用户 60 天内必须修改账号密码。

```
#passwd -x 60 user1
```

【示例】设置在 user1 账户密码必须修改前 3 天向该用户发送警告信息。

```
#passwd  -w 3  user1
```

【示例】将 user1 账户的密码策略还原为默认设置。

```
#passwd  -n 0 –x 99999 -w 7  user1
```

5.5 使 用 账 户

5.5.1 账户的查询操作

1. 查询登录历史

当需要查询当前与过去都有哪些用户登录过系统时，可以使用 last 命令。

last 命令用于显示当前和过去登录系统的用户列表。

下面对 last 命令的用法加以说明。

【语法】**last**

```
[root@hero ~]# last
user1    tty1                        Tue Mar 29 13:11    still logged in
user1    pts/2       192.168.1.104   Tue Mar 29 13:10    still logged in
root     pts/3       192.168.1.104   Tue Mar 29 10:27    still logged in
root     pts/2       192.168.1.104   Mon Mar 28 14:31 -  09:47  (19:15)
root     pts/1       :0.0            Mon Mar 28 14:31    still logged in
…
```

其输出格式为：

账号名称　登录终端　登录地址　登录时间信息　登录状态

last 命令是根据/var/log/wtmp 日志文件的结果进行输出的。从输出的结果中可以查看到哪些用户在什么时间、什么地点登录了系统，持续时间有多长等一系列信息。

2. 查询每个账号的最近登录时间

当需要知道每个账号最后一次登录系统的时间时，可以使用 lastlog 命令。loastlog 命令会去读取/var/log/lastlog 文件，并将结果数据输出。

下面对 lastlog 命令的用法加以说明。

【语法】**lastlog**

```
[root@hero ~]# lastlog
Username         Port     From             Latest
root             pts/3    192.168.1.104    Tue Mar 29 10:27:36 +0800 2011
bin                                        **Never logged in**
daemon                                     **Never logged in**
```

```
adm                                        **Never logged in**
…
```

3. 显示当前已登录到系统中的所有用户

当需要知道当前有哪些用户登录到系统中，各自执行了哪些操作的时候，可以使用 w 命令。

下面对 w 命令的用法加以说明。

【语法】**w**

```
[root@hero ~]# w
 00:43:05 up  4:38,  5 users,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM              LOGIN@  IDLE    JCPU    PCPU WHAT
user1    tty1      -                29Mar11 276days 0.03s   0.03s -bash
root     :0       -                 28Mar11 ?xdm?   48.68s  0.49s
                                                    /usr/bin/gnome-session
root     pts/1    :0.0              28Mar11 276days 0.07s   0.07s bash
user1    pts/2    192.168.1.104     29Mar11 1:51    0.09s   0.09s -bash
root     pts/3    192.168.1.104     29Mar11 0.00s   1.30s   0.02s w
```

利用 w 命令也可以查看特定用户的登录状态，使用“w 用户名”命令。

```
[root@hero ~]# w root
 00:43:42 up  4:38,  5 users,  load average: 0.06, 0.02, 0.00
USER     TTY      FROM              LOGIN@  IDLE    JCPU    PCPU WHAT
root     :0       -                 28Mar11 ?xdm?   48.72s  0.49s
                                                    /usr/bin/gnome-session
root     pts/1    :0.0              28Mar11 276days 0.07s   0.07s bash
root     pts/3    192.168.1.104     29Mar11 0.00s   1.29s   0.01s w root
```

与 w 命令类似的还有 who 命令，但 who 命令仅显示账号名称、登录地点和登录时间等信息，比 w 命令略为简略。

who 命令用于显示登录到系统中的所有用户。

下面对 who 命令的用法加以说明。

【语法】**who**

```
[root@hero ~]# who
user1    tty1         Mar 29 13:11
root     :0           Mar 28 14:31
root     pts/1        Mar 28 14:31 (:0.0)
user1    pts/2        Mar 29 13:10 (192.168.1.104)
root     pts/3        Mar 29 10:27 (192.168.1.104)
```

who am i 命令作为 who 命令的特殊格式用于显示当前用户的信息。

【语法】**who am i**

```
[root@hero ~]# who am i
root     pts/3        Mar 29 10:27 (192.168.1.104)
```

whoami 命令用于显示当前用户的用户名。

【语法】**whoami**

```
[root@hero ~]# whoami
root
```

5.5.2 账户的检查工具

当完成账户的添加和删除等操作之后，可以利用一系列工具来检查用户组、账号和安全口令等配置文件的一致性，防止因不同步的配置或手工操作导致账户错误。

1. pwck

pwck 命令用于检查/etc/passwd 账号配置文件内的信息，以及实际的用户主目录是否存在等信息，还可以比较/etc/passwd 和/etc/shadow 的信息是否一致，另外，如果/etc/passwd 内的数据字段存在错误，会提示用户修改。

```
[root@hero ~]# pwck
user adm: directory /var/adm does not exist
user news: directory /etc/news does not exist
user uucp: directory /var/spool/uucp does not exist
user gopher: directory /var/gopher does not exist
user ftp: directory /var/ftp does not exist
user pcap: directory /var/arpwatch does not exist
user avahi-autoipd: directory /var/lib/avahi-autoipd does not exist
pwck: no changes
```

这里，pwck 提示这些用户的用户主目录不存在。

2. chpasswd

chpasswd 命令的作用是读入未加密的密码，经过加密后将加密密码写入/etc/shadow 文件中。

该命令的用法是：

【语法】**#echo "用户名：密码" | chpasswd**

经过了上述一系列的配置后，有关用户账户和用户组的基本知识就介绍完了。用户管理工作是系统管理员的一个很重要的工作内容。本章内容较多，连续性较强，需要大家认真掌握。

在充分理解和掌握本章的内容之后，下一章将开始目录和文件系统的学习。读者会发现，当初还让人感觉很陌生的这个操作系统变得越来越丰富多彩了。

第 6 章 文件与目录系统

文件与目录系统是 Linux 系统中很重要的一种控制结构。本章内容以实用性及概念为主，读者应先熟悉基本的文件与目录操作，再逐渐深入了解系统的结构与深层的概念，以期对该系统能运用自如。

6.1 目录与文件基础

在学习 Linux 的过程中经常会在不同场合使用“文件系统”这个词，文件系统一词可以代表以下两种含意：

（1）文件系统是磁盘分区（partition）和文件在磁盘中存储的逻辑结构的总称。

（2）磁盘分区或其他存储设备中包含的目录树也称为文件系统。

6.1.1 查看文件与目录

在登录系统后，可以查看当前工作目录下的文件与子目录，使用命令：

```
# ls  -l
```

会看到如下格式的显示：

```
[root@hero ~]# ls -l
total 112
drwxr-xr-x  3   root root   4096    Mar 16  2011 Desktop
-rw-------  1   root root   1250    Mar 14  2011 anaconda-ks.cfg
-rw-r--r--  1   root root  12866    Mar 29  2011 file1
-rw-r--r--  1   root root      0    Mar 29  2011 file2
-rw-r--r--  1   root root  29387    Mar 14  2011 install.log
-rw-r--r--  1   root root   4434    Mar 14  2011 install.log.syslog
-rw-r--r--  1   root root   3544    Mar 29  2011 lftp.conf
drwxr-xr-x 14   root root   4096    Mar 29  2011 log
drwxr-xr-x  2   root root   4096    Mar 29  2011 lpi
-rw-------  1   root root   9285    Mar 25  2011 mbox
drwxr-xr-x  9   root root   4096    Mar 25  2011 test
drwxr-xr-x  2   root root   4096    Mar 29  2011 xinya
```

上面的命令以长格式的方式显示文件与目录，每一行都是一个文件或子目录的属性

数据，每个文件或子目录的属性数据又以数个字段显示，各字段说明如下：

```
-rw-------   1    root    root    1250    Mar 14  2011  anaconda-ks.cfg
 ↑           ↑     ↑       ↑        ↑          ↑               ↑
文件类型    文件的  文件    文件   文件的容   文件的创建        文件的名称
与权限      连接数  属主    属组    量（B）     时间
```

（1）文件类型与权限：该字段共有 10 个字符，第一个字符表示文件的类型，剩下的 9 个字符表示文件的权限状态。有关权限的内容将在随后的章节讨论，这里重点说明第 1 个字符所代表的文件类型。第 1 个字符所代表的文件类型如表 6.1 所示。

表 6.1　文件类型

字符	类　　型	字符	类　　型
-	普通文件	s	socket 套接字文件
d	目录	p	命名管道 FIFO 文件
b	块专用文件（Block-Special File）	l	符号链接文件
c	字符专用文件（Character-Special File）		

根据表 6.1，通过文件类型与权限字段第一个字符就可以判断一个文件的类型，如：

```
-rw-r--r--  1  root  root  29387  Mar 14  2011  install.log
```

“-”代表 install.log 是一个普通文件。

```
[root@hero ~]# ls -l /dev/hda
brw-r-----  1  root  disk  3, 0  Dec 31 16:37  /dev/hda
```

“b”代表/dev/hda 是一个块设备文件（硬盘）。

```
[root@hero ~]# ls -l /dev/tty1
crw-------  1  root  root  4, 1  Dec 31 16:39  /dev/tty1
```

“c”代表/dev/tty1 是一个字符设备文件。

```
 [root@hero ~]# ll
…
drwxr-xr-x  2  root  root  4096  Mar 29  2011  xinya
```

“d”代表 xinya 是一个目录。

（2）连接数：表示该文件或目录所建立的连接的数量。

（3）属主：表示拥有该文件或目录的用户账号。

（4）属组：表示拥有该文件或目录的组账号。

（5）文件的容量：默认以 byte（B）为单位进行计算，表示该文件的大小。

（6）创建时间：创建这个文件或目录的日期和时间。

（7）文件名：文件或目录的名称。

6.1.2 文件与目录名称

文件或目录在 Linux 中的命名规则很简单，以字母、数字或符号组成，中间不能使用空格，而文件名称的长度最长可以达到 256 个字符，但考虑到与其他版本 Linux 或其他操作系统的兼容性，一般建议将文件名限制在 14 个字符以内。

文件名或目录名以“.”符号来分隔出扩展名，如 myfile.txt。在一般的使用习惯下，特定类型的文件会使用扩展名标示；然而对 Linux 系统而言，一个文件的类型是按其内容决定的，扩展名只是方便用户标识文件。前面已经介绍过，判断文件类型的方法是使用 file 命令。

此外，有一些符号字符对 Shell 有特殊意义，应该尽量避免在文件名中使用。这些符号中有一些是通配符，有一些是具有特殊含义的转义字符，如*、/、\、[]、()、？和$等符号。

6.1.3 管理权限与所属用户和组

Linux 是一个很重视安全性的操作系统，因为允许多人登录系统，所以对目录或文件进行访问及执行的权限就很重要。管理员需要通过不同的权限划分来控制不同用户对系统资源的使用范围。

1. 查看权限

以“ls -l”命令长格式显示文件及子目录时，第一个字段除了表示文件类型的第 1 个字符外，其余 9 个字符用来显示文件的权限状态。

```
- r w - r - - r - -  1  root  root  29387  Mar 14  2011  install.log
        ↑
 9 位字符表示文件权限
```

9 个字符分为 3 组，每组包括 3 个字符。第一组为属主权限，第二组为属组用户权限，第三组为其他用户权限。

```
-    -    -    |    -    -    -    |    -    -    -
↑    ↑    ↑         ↑    ↑    ↑         ↑    ↑    ↑
读   写   执行      读   写   执行      读   写   执行
r    w    x         r    w    x         r    w    x
2^2  2^1  2^0       2^2  2^1  2^0       2^2  2^1  2^0
  属主权限            属组权限            其他用户权限
```

每组权限所代表的含义如上，第一位代表是否具有读权限，如果有则以“r”表示，如果没有则以“-”表示；第二位代表是否具有写权限，如果有则以“w”表示，如果没有则以“-”表示；第三位代表是否具有执行的权限，如果有则以“x”表示，如果没有则以“-”表示。

根据以上规则，可以限定属主、属组和其他用户对文件或目录的权限。实例如表 6.2

所示。

表 6.2　文件权限实例

字　　符	权　　限
drwxr-xr-x	d 表示是一个目录，属主有读取、写入和进入该目录的权限，属组用户和其他用户有读取和进入该目录的权限
-rw-r--r--	属主可以读写该文件，属组用户和其他用户可以读取该文件
-rw-------	属主用户可以读写该文件，属主用户和其他用户对该文件没有任何权限
-rwxr-xr-x	属主用户可以读取、写入和执行该文件，属组用户和其他用户对该文件具有读取和执行的权限

读者在了解了权限结构后可能会产生一个问题，那就是“一个文件的属主和属组是谁呢？”这一点在文件和目录的属性数据中有明确的表示。

```
-rw-------    1    root    root    1250    Mar 14  2011    anaconda-ks.cfg
    ↑         ↑     ↑       ↑        ↑          ↑                 ↑
文件类型    文件的  文件    文件    文件的容    文件的创建       文件的名称
与权限      连接数  属主    属组     量（B）      时间
```

上例中的 anaconda-ks.cfg 文件的属主为 root 用户，属组为 root 组。

2. 修改文件或目录的权限

系统管理员或文件、目录的用户可以根据需要改变文件或目录的权限。再配合组的管理，就可以指定文件或目录以不同的权限开放给不同的对象。修改文件的权限需要使用 chmod 命令。只有文件或目录的属主用户或 root 能改变权限。chmod 命令有两种工作方式，分别说明如下。

【语法 1】**chmod [who] opcode　permission　file1 [file2 …]**

对参数与选项说明如下：

who 参数：表示修改哪个用户的权限，此处可使用的用户标识包括：

u：表示属主。

g：表示属组。

o：表示其他用户。

a：表示所有用户。

opcode：操作代码，表示对权限如何修改，此处可使用的操作代码包括：

+：表示增加权限，即在原有权限上增加权限。

–：表示删除权限，即从原有权限中减少权限。

=：表示分配权限，同时将原有的权限删除。

permission：操作权限，表示要修改何种权限，此处可定义的权限包括：

r：读取。

w：写入。

x：执行。

file 文件列表：要改变权限的文件夹或目录名称。可以一次改变多个文件或目录，文件或目录名称之间用空格分开。

【示例】将 install.log 文件的权限修改为属组用户与其他用户可读取和写入。

```
-rw-r--r--  1 root root 29387 Mar 14  2011 install.log
```

该操作实际上是为属组用户和其他用户添加写入权限。

```
[root@hero ~]# ls -l install.log
- r w - r - - r - - 1 root root 29387 Mar 14  2011 install.log
[root@hero ~]# chmod g+w,o+w install.log
[root@hero ~]# ls -l install.log
- r w - r w - r w - 1 root root 29387 Mar 14  2011 install.log
```

【语法 2】**chmod　权限代码　文件或目录名列表**

这里的权限代码是将属主、属组和其他用户的权限按位添加位权值以得到最终的权限。

```
-    -    -   |   -    -    -   |   -    -    -
↑    ↑    ↑       ↑    ↑    ↑       ↑    ↑    ↑
读   写   执行     读   写   执行     读   写   执行
r    w    x       r    w    x       r    w    x
2²   2¹   2⁰      2²   2¹   2⁰      2²   2¹   2⁰
  属主权限           属组权限          其他用户权限
```

如属主的权限为可读、可写、可执行，则权限代码为 $2^2\times1+2^1\times1+2^0\times1=4+2+1=7$。属组的权限为可读、可写，则权限代码为 $2^2\times1+2^1\times1+2^0\times0=4+2+0=6$。其他用户的权限为可读，则权限代码为 $2^2\times1+2^1\times0+2^0\times0=4+0+0=4$。这时该文件的权限代码为 764。

```
[root@hero ~]# ls -l anaconda-ks.cfg
-rw------- 1 root root 1250 Mar 14  2011 anaconda-ks.cfg
[root@hero ~]# chmod 764 anaconda-ks.cfg
[root@hero ~]# ls -l anaconda-ks.cfg
-rwxrw-r-- 1 root root 1250 Mar 14  2011 anaconda-ks.cfg
```

chmod 命令可以使用-R 选项，-R 选项的作用是将目录下的子目录和文件一并修改。

3．修改属主和属组

当用户 user1 创建一个文件或目录时，系统默认这个文件或目录的属主和属组均为 user1。系统管理员 root 和文件或目录的属主用户有权修改文件或目录的属主和属组。修改文件的属主可以使用 chown 命令，修改文件的属组可以使用 chgrp 命令。

chown 命令可以改变文件或目录的所有权与属主用户，也可以改变文件的属组。与 chgrp 命令一样，chown 可以同时改变数个文件或目录的所有权。只有 root 用户或文件的属主用户有权使用 chown 命令。

下面对 chown 命令的用法加以说明。

【语法】**chown 新属主[.新属组] file1 [file2…]**

【示例】将 test2 文件的属主修改为 root 用户。

```
[root@hero ~]# ls -l test2
-rw-r--r-- 1 user1 root 0 Dec 31 17:21 test2
[root@hero ~]# chown root test2
[root@hero ~]# ls -l test2
-rw-r--r-- 1 root root 0 Dec 31 17:21 test2
```

chgrp 命令可以改变文件或目录的属组，且可以同时改变多个文件或目录的属组。文件或目录的属主用户或 root 才有使用 chgrp 命令的权利。

chgrp 命令说明如下。

【语法】**chgrp 新属组 file1 [file2 …]**

【示例】将 test2 文件的属组修改为 root 组。

```
[root@hero ~]# ls -l test2
-rw-r--r-- 1 user1 user1 0 Dec 31 17:21 test2
[root@hero ~]# chgrp root test2
[root@hero ~]# ls -l test2
-rw-r--r-- 1 user1 root 0 Dec 31 17:21 test2
```

chgrp 命令可以使用-R 选项，-R 选项的作用是将目录下的子目录和文件一并修改属组属性。

【示例】将 test2 文件的属主和属组均修改为 user1。

```
[root@hero ~]# ls -l test2
-rw-r--r-- 1 root root 0 Dec 31 17:21 test2
[root@hero ~]# chown user1.user1 test2
[root@hero ~]# ls -l test2
-rw-r--r-- 1 user1 user1 0 Dec 31 17:21 test2
```

6.1.4 专门用户组配置法

在 CentOS 5.X 中使用专门用户组（UPG，User Private Group）配置法，让系统管理员能方便管理组及文件的权限。专门用户组配置法（user private group scheme）有下列作用。

1. 创建个人组（UPG）

以 useradd 命令添加用户账号时，若不指定属组时，系统会默认创建一个用户自己的原始组，该账号就是原始组的唯一成员。

2. 改变文件的组保护（Group Protection）

在一般 UNIX 系统的设置中，用户文件只有所有者才能修改，而即使同一个原始组

的其他用户也没有修改的权限，这就是文件的组保护。在 CentOS 的 UPG 配置法下，每个用户有自己的原始组，所以这样的组保护就不需要了。这一项设置在/etc/profile 文件中：umask=002。

3. 设置目录的属组

对一个目录设置属组，同时设置组标识符，之后在这个目录下产生的文件，其所属组会与目录相同。也就是说，对于在这个目录下产生的所有文件，目录的所属组成员有相同的访问权限。

4. 设置工作组

如果在系统中有一个工作计划，名称为 bc-project，系统内用户参与该工作的成员有 user1、user2 和 user3。这 3 个成员必须对 bc-project 相关的文件有完整的访问权限，所以这 3 个用户可以另外组成一个组，如图 6.1 所示。

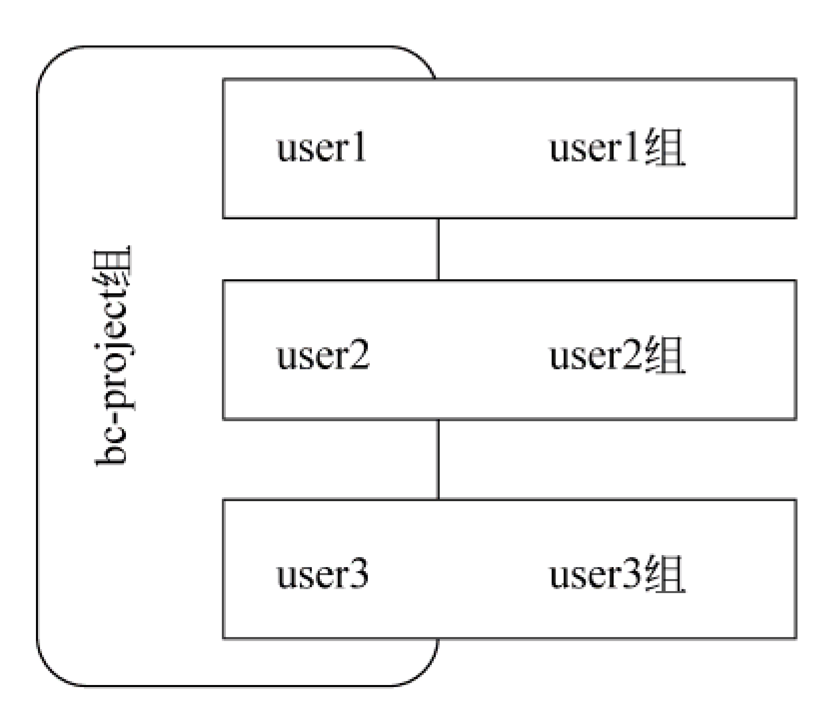


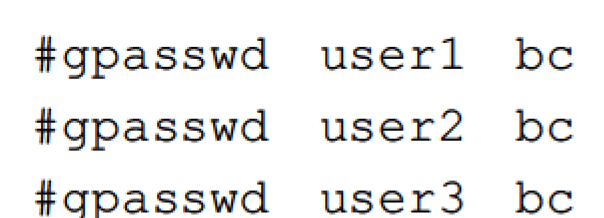
图 6.1　bc-project 工作组

系统管理员 root 可以使用下面的步骤对工作组进行设置：

（1）设置工作组。首先为工作计划 bc-project 创建工作组，组名称为 bc：

```
# groupadd  bc
```

（2）加入组成员。将参与计划的组成员加到 bc 组内成为 bc 的成员：



```
#gpasswd  user1  bc
#gpasswd  user2  bc
#gpasswd  user3  bc
```

（3）设置工作目录。先创建计划的目录 projects，在计划目录下创建 bc-project 的工作组目录：

```
#mkdir  -p  /profects/bc-project
```

然后设置工作目录的属组为 bc：

```
#chgrp  -R  bc  /profects/bc-project
```

（4）设置目录权限。将工作组对这个目录的权限设置成完整权限。并设置 GID：

```
#chmod  2775  /profects/bc-project
```

这样，bc 组的成员都能在 bc-project 目录下新建文件，而且所创建文件的属组为 bc，每个成员对这些文件都会有完整的访问和编辑的权限。

6.2 文件与目录属性的默认值

6.2.1 文件的默认权限

在 Linux 中，创建文件时的默认权限为 644(rw-r--r--)，而目录的默认权限为 755(rwxr-xr-x)。这个默认权限的产生及其值取决于系统的 umask 值。

umask 值是一个权限的掩码，系统根据该掩码来定义文件和目录的默认权限。可以使用 umask 命令来查看该掩码。

```
[root@hero ~]# umask
0022
[root@hero ~]# umask -S
u=rwx,g=rx,o=rx
```

查看方式有两种：一种是直接输入 umask 命令，可以看到数字类型的权限设置值；另一种是使用“umask -S”，会以符号的方式显示默认权限。注意，在“umask -S”命令的输出结果中，如果是文件则不考虑 x 权限，文件的默认权限为“u=rw，g=r，o=r”。

对于“umask -S”的输出结果，由于比较直观易读，这里不做过多的讨论。下面要详细讨论 umask 命令的输出结果 0022。

umask 命令的输出结果 0022 实际上是一个掩码，其表示的权限位如图 6.2 所示：

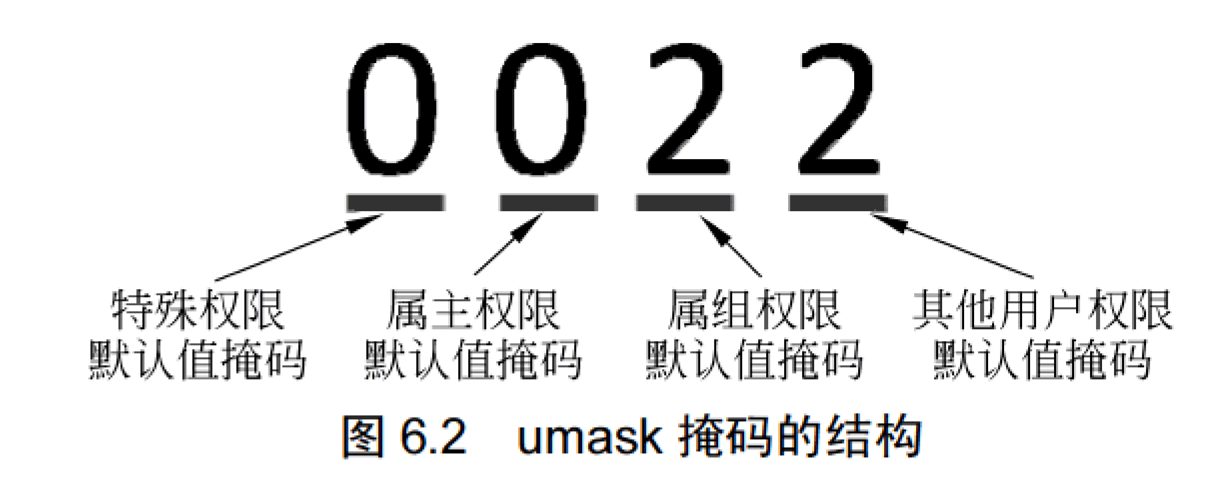


图 6.2 umask 掩码的结构

第 1 位表示的是特殊权限的默认掩码，有关于特殊权限将在 6.2.2 节介绍。

第 2 位表示的是属主权限的默认掩码，即属主默认的权限。

第 3 位表示的是属组权限的默认掩码，即属组默认的权限。

第 4 位表示的是其他用户权限的默认掩码，即其他用户默认的权限。

这里先不考虑第 1 位特殊权限掩码，先讨论后 3 位掩码。umask 定义的是默认权限的掩码。所谓掩码是真值的取反值。下面使用两个例子来说明掩码。

（1）文件默认权限的掩码值。文件在定义默认权限时是不考虑可执行权限的，因为文件是否可执行是文件的一种功能性的体现，需要权限和文件内容的共同定义，所以，默认的情况先不考虑文件的可执行权限问题。文件的默认权限掩码值结构如图 6.3 所示。由图 6.3 可见，umask 文件掩码是将属主、属组和其他用户的权限在不考虑 x（可执行）权限的情况下按位取反后得到的权限值。

（2）目录的默认权限值掩码。与文件不同，目录在定义默认权限时是需要考虑可执

行权限的。目录的可执行权限的含义是该用户是否有进入该目录的权限。目录的默认权限掩码值结构如图 6.4 所示。

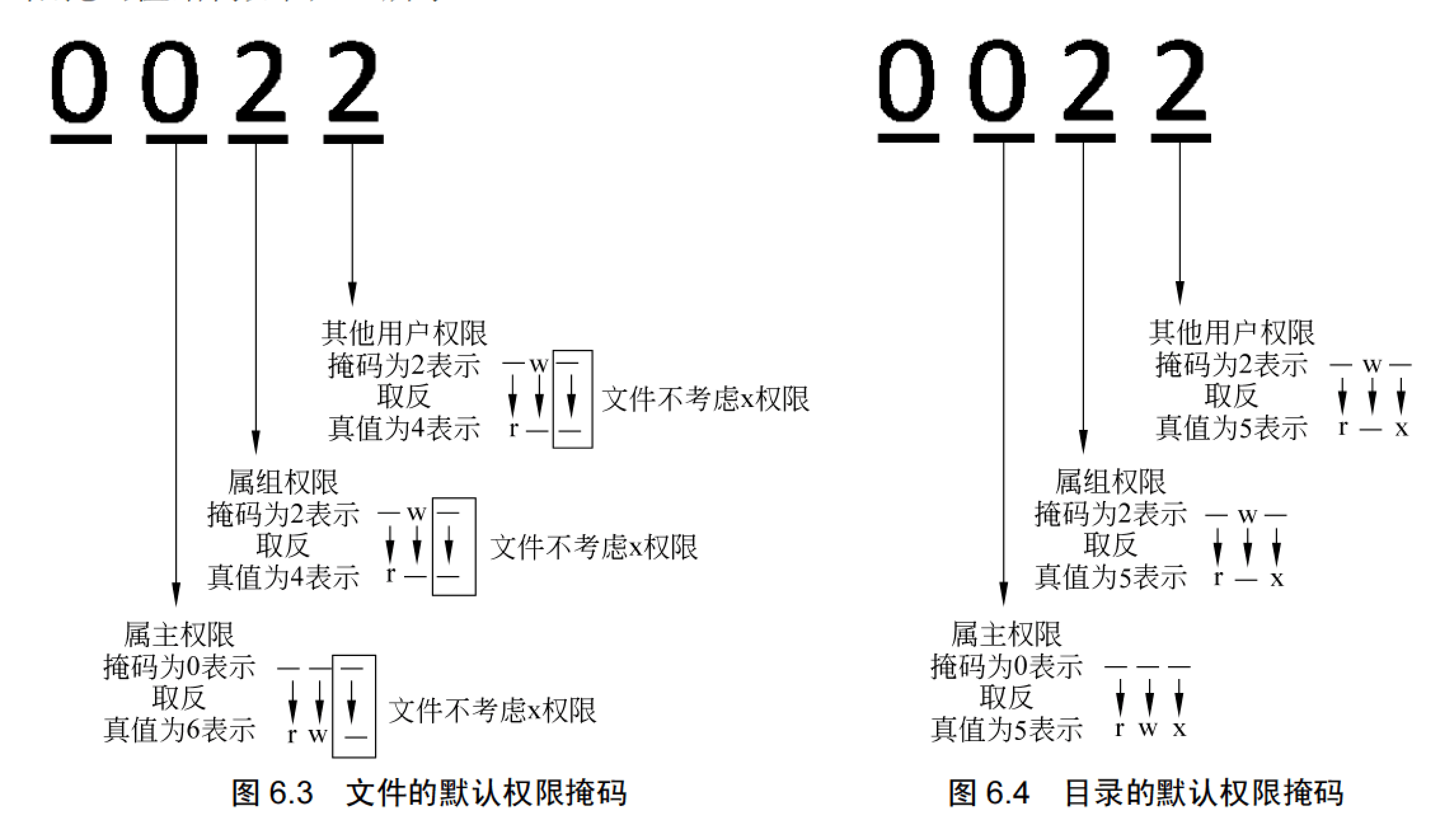


图 6.3　文件的默认权限掩码　　图 6.4　目录的默认权限掩码

由图 6.4 可见，umask 目录掩码是将属主、属组和其他用户的权限按位取反后得到的权限值。

通过以上的分析，应该对前面所给出的定义“所谓掩码是真值的取反值”有所认识了，同时也应该明确文件和目录的默认权限的来源为何了。

在了解了 umask 的含义与功能之后，当需要修改 umask 掩码以适应用户对默认权限的特殊要求时，可以使用“umask 掩码值”这个命令来实现。

【示例】设置 umask 掩码，使得创建文件的默认权限为 r--------，400。

注意，如果要求权限的默认值为 400，则 umask 的掩码应为 0266，如图 6.5 所示。

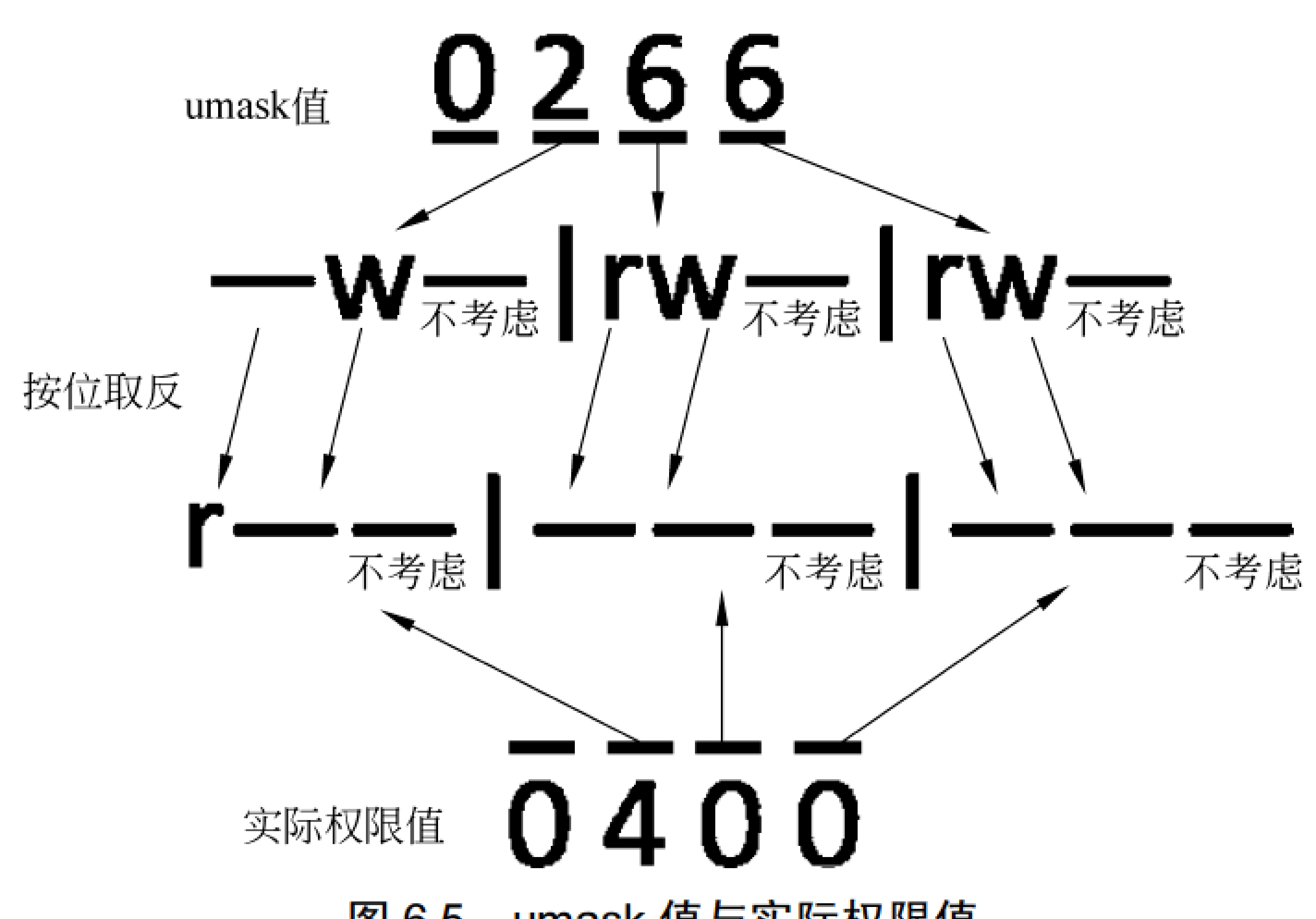


图 6.5　umask 值与实际权限值

```
[root@hero ~]# umask
0022                                                  ←umask 的默认值为 0022
[root@hero ~]# touch file1
[root@hero ~]# ls -l file1
-rw-r--r-- 1 root root 12866 Dec 31 17:28 file1 ←创建文件的默认权限为 644
[root@hero ~]# umask 0266                             ←设置 umask 值为 0266
[root@hero ~]# touch file2
[root@hero ~]# ls -l file2
-rw------- 1 root root 0 Dec 31 17:28 file2     ←创建文件的默认权限为 400
```

注意，当用户使用 umask 命令重新设置了 umask 值后，在用户重新登录系统后，umask 值依旧会恢复为 0022。如果需要固定 umask 的自定义值，需要在环境变量配置文件中定义。有关环境变量配置文件将在后续章节中介绍。这里可以先将 umask 的定义值写入 /etc/bashrc(将对所有用户有效)或~/.bashrc(仅对当前用户有效)，可以在这两个文件的末尾添加“umask=自定义值”。

6.2.2 文件的特殊权限

文件的重要权限就是 r（读）、w（写）和 x（执行）这 3 个权限，除这 3 种权限外，还具有 3 种特殊的权限 SUID、SGID 和 SBit，这 3 种权限为文件的访问控制提供了更为灵活的应用。以下将详细阐述这 3 种权限的作用。

1. Set UID

Set UID 简写为 SUID，一个文件如果具有该权限，则普通用户在执行该文件时将具有该文件属主的权限。

该权限借用文件属主的 x 位表示，一旦文件被设置为 Set UID，则文件权限中属主的 x 位将变为 s；如果文件权限中属主的 x 未被设置，则使用 S(大写字母)，如：

```
[root@hero ~]# ls -l /bin/ping
- r w s r - x r - x  1 root root 35832 Sep 27  2009 /bin/ping
                                                  ←权限中含有“s”
```

Set UID 的作用是，对于一些可执行文件而言，需要将其使用权限开放给普通用户，如 ping。但如果该程序需要调用一些只有管理员可使用的系统功能时，由于是以普通用户身份执行的该应用程序，所以应用程序可能会出现执行错误的情况。为解决这一问题，可以为应用程序赋予 SUID 的权限。

SUID 的应用一方面有利于应用程序更灵活地调用系统丰富的功能，另一方面会造成系统用户特别是 root 用户权限的泄露。所以当前系统管理中的趋势是逐渐减少 SUID 权限的使用。

2. Set GID

Set GID 简写为 SGID，与 SUID 类似，一个文件如果具有该权限，则普通用户在执

行该文件时将具有该文件属组的权限。

SGID 可以应用在两个方面。

（1）应用于文件：如果 SGID 设置在二进制文件上，则不论用户是谁，在执行该程序时，它的有效用户组为该程序的属组。

（2）应用于目录：如果 SGID 设置在目录上，则在该目录中建立的文件或子目录的属组将会是该目录的属组。

该权限借用文件或目录的属组权限中的 x 位表示。一旦文件被设置 Set GID，则文件或目录权限中属组的 x 位将变为 s；如果文件或目录权限中属组的 x 未被设置，则使用 S（大写字母）。

SGID 往往应用于合作项目所属的文件或目录中，在系统管理中很少使用。

3. Sticky Bit

Sticky Bit 简写为 SBit，当前只针对目录有效，对文件没有效果。SBit 权限的作用是，在具有 SBit 权限的目录下，用户如果对该目录有 w 和 x 权限，则当用户在该目录下建立文件或目录时，只有文件的属主和 root 才有权利删除。

4. 设置特殊权限

特殊权限在普通权限之外使用 3 位来表示，如图 6.6 所示。

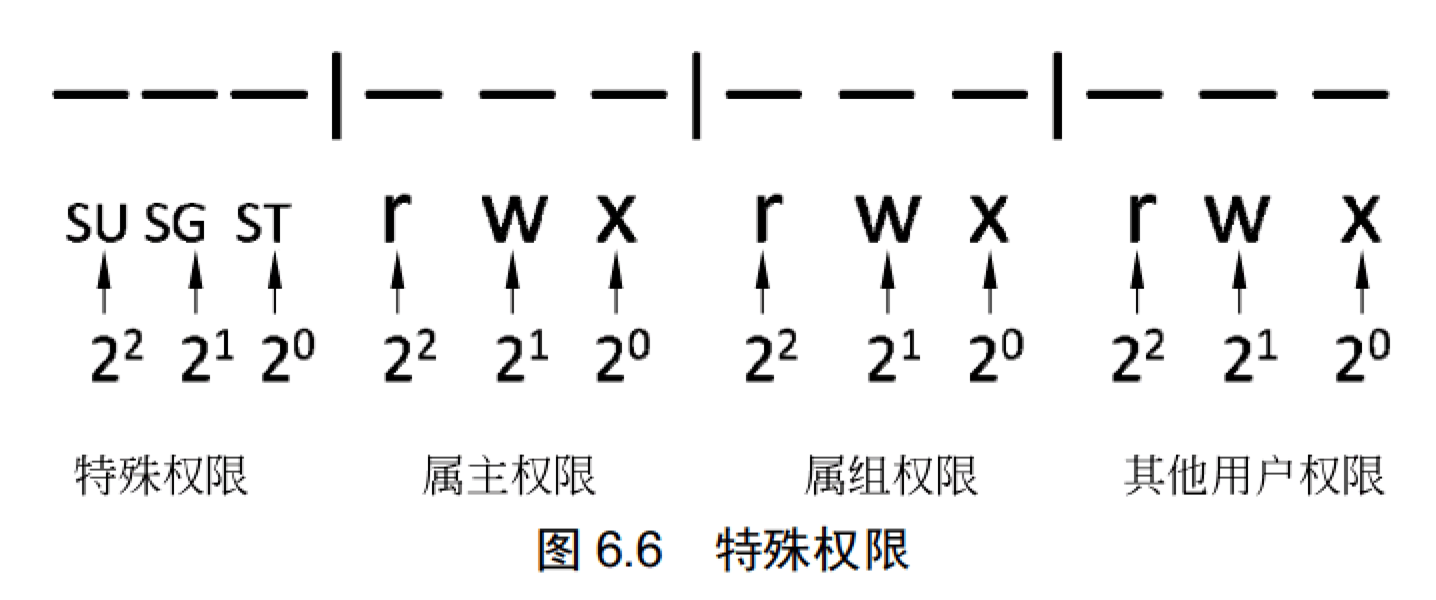


图 6.6　特殊权限

根据图 6.5 所示的权限位分布情况，当需要设置特殊权限值时，在普通权限值前添加特殊权限值即可。如设置一个权限为 644 的文件的特殊权限为 SUID、SGID，则可以使用 6644，即在普通权限值 644 前添加特殊权限 SUID2^2+SGID2^1 为 6，得到 6644。

```
[root@hero ~]# ls -l file1
-rw-r--r--1 root root 12866 Dec 31 17:28 file1 ←普通文件权限为 644
[root@hero ~]# chmod 6644 file1
[root@hero ~]# ls -l file1
-rwSr-Sr--1 root root 12866 Dec 31 17:28 file1
```

也可以使用符号的方法配置特殊权限。上例可以使用：

```
[root@hero ~]# ls -l file1
-rw-r--r--1 root root 12866 Dec 31 17:28 file1
[root@hero ~]# chmod u+s,g+s file1
```

```
[root@hero ~]# ls -l file1
-rwSr-Sr--1 root root 12866 Dec 31 17:28 file1
```

前面在介绍 umask 时，其默认值为 0022，第一个 0 表示的是特殊权限的默认设置。当前已很少再使用 umask 定义文件的特殊权限了。

6.2.3 目录属性的意义

对于文件属性中的读、写和执行等权限，其含义比较好理解。但是对于目录而言，读、写和执行究竟包含有哪些可提供的操作呢？

（1）目录的读属性：表示具有读取目录结构清单的权限，使用 ls 命令可以将该目录中的文件和子目录的内容列出来。

（2）目录的写属性：表示具有更改目录结构清单的权限，包括以下操作：

① 建立新的文件与目录。

② 删除已经存在的文件与目录（不论该目录的属主是谁）。

③ 重命名目录和文件。

④ 移动目录中文件和子目录的位置。

（3）目录的执行权限：表示具有进入该目录的权限。

6.2.4 文件的隐藏属性

文件有隐藏属性，隐藏属性对系统有很重要的作用，尤其是在系统安全性方面非常重要。

1. 设置文件的隐藏属性

设置文件的隐藏属性需要使用 chattr 命令。

下面对 chattr 命令的用法加以说明。

【语法】**chattr [+–=] [AacdijSsu] 文件或目录名**

选项说明如下：

+：为文件添加隐藏属性。

–：为文件删除隐藏属性。

=：为文件重新赋予隐藏属性。

A：文件或目录的访问时间不能被修改。

a：文件中只能增加数据，不能删除。

c：文件将被自动压缩，读取时自动解压缩。

d：在 dump 备份时，文件或目录具有自动转储功能。

i：设置文件为只读，不能被删除、改名、链接和写入等。

j：在 ext3 文件系统中文件将被采用日志式操作。

S：类似于 sync，具有数据同步写入功能。

s：删除文件时，将文件从磁盘中彻底删除，不能恢复。

u：与 s 相反，删除时，文件不从磁盘中清除，有利于恢复。

【示例】为文件设置只读属性。

#chattr +i 文件名

```
[root@hero ~]# ll file1
-rw-r--r-- 1 root root 12866 Dec 31 17:28 file1
[root@hero ~]# chattr +i file1
[root@hero ~]# rm file1
rm: remove write-protected regular file 'file1'? y
rm: cannot remove 'file1': Operation not permitted
```

【示例】删除文件的只读属性。

#chattr -i 文件名

```
[root@hero ~]# chattr -i file1
[root@hero ~]# rm file1
rm: remove regular file 'file1'? y
[root@hero ~]#
```

2. 查看文件的隐藏属性

文件的隐藏属性不会通过“ls -l”命令显示出来，当需要查看时要使用 lsattr 命令。

lsattr 命令用于显示文件的隐藏属性。

下面对 lsattr 命令的用法加以说明。

【语法】**lsattr [-aR] 文件或目录名**

选项说明如下：

-a：将隐藏文件的属性也显示出来。

-R：连同子目录的数据一并显示出来。

```
[root@hero ~]# lsattr file1
----ia-A-j--- file1
```

6.2.5 文件的时间戳信息

在了解了文件的类型以及权限设置之后，还需要了解文件的时间信息。文件的时间信息在当前的学习内容中好像是作用不大，但是在今后的系统管理中，特别是日志管理中的作用还是很重要的，所以对于文件的时间信息应有一个系统的认识。

对于一个文件而言，会存在以下 3 个主要的事件标志。

（1）modification time（mtime，修改时间）：当文件的内容数据发生变化时会更新这个时间。该时间反映了文件何时被修改。注意，是文件的内容发生变化而不是文件的属性发生变化。

（2）status time（ctime，状态时间）：当文件的状态发生改变时会更新该时间，文件的状态包括权限状态和属性状态等。

（3）access time（atime，访问时间）：当读取文件内容时会更新该时间。

默认的情况下，当使用“ls -l”命令时，显示的是文件的修改时间 mtime。对于文件的时间信息的查看可以使用 ls 命令的“--time”选项。

下面对 ls 命令的 time 选项加以说明。

【语法】**ls --time=atime | ctime [文件名]**

【示例】查看文件的修改时间 mtime。

```
[root@hero ~]# ls -l install.log
-rw-rw-rw- 1 root root 29387 Mar 14  2011 install.log
                                    ↑
                          默认显示文件的修改时间
```

【示例】查看文件的访问时间。

```
[root@hero ~]# ls -l --time=atime install.log
-rw-rw-rw- 1 root root 29387 Dec 31 17:40 install.log
                                    ↑
                               文件的访问时间
```

【示例】查看文件的状态修改时间 ctime。

```
[root@hero ~]# ls -l --time=ctime install.log
-rw-rw-rw- 1 root root 29387 Dec 31 17:17 install.log
                                    ↑
                             文件的状态修改时间
```

当文件的时间戳信息需要被修改时可以使用 touch 命令。

下面对 touch 命令的用法加以说明。

【语法】**touch [选项] 文件名 …**

选项说明如下：

-a：仅修改访问时间 atime。

-c：仅修改时间而不建立文件。

--date=：修改时间。

-m：修改 mtime。

-t：后面可以接时间，格式为 YYMMDDhhmm。

6.3 目录与文件系统

6.3.1 Linux 的标准文件系统

在每个 UNIX 系统，文件目录都有一个规范，不管用户登录世界上的哪一台 UNIX 主机，都可以知道系统中哪一类文件会放在什么目录下。在 Linux 中也有这样的规范，遵循 Linux 文件系统标准，每一台 Linux 主机都有相似的文件系统。这个规范就是文件

系统层次标准（FHS）。

FHS 是一个指南，由对类 UNIX 操作系统下的文件和目录放置的要求和规定组成。这些规定能够较好地支持应用软件、系统管理员工具、开发工具及脚本之间的互操作以及这些系统的文档一致性。FHS 定义了根目录下第一层目录的分布和每个目录所应包含的内容，同时，对/usr 和/var 目录中的个别子目录也进行了定义。本节介绍常见的目录含义，对其更详细的了解需要在以后的学习过程中逐步积累。

6.3.2 Linux 系统中重要的标准目录和文件

1. /（根目录）

根目录位于文件系统的最顶层，用斜线（/）表示。它包含了所有的目录和文件。

2. /bin

/bin 目录也称为二进制目录，它包含了那些提供给系统管理员和普通用户使用的命令。该目录下的文件，有的是可执行文件，有的是其他目录下可执行文件的符号连接。常用的命令基本都在该目录下，如 bash、cat、chmod 和 ls 等。

3. /boot

/boot 目录存放了用于启动 Linux 系统的所有文件，如内核文件和 grub 的配置文件。内核的文件名一般是 vmlinuxz 加上版本号。

4. /dev

/dev 目录也称为设备目录。该目录存放连接到计算机上的设备（光驱、硬盘和打印机等）的对应文件。这些文件被称为特殊文件。特殊文件分为两种：块特殊文件和字符特殊文件。字符特殊文件对应键盘等面向字符 I/O 操作的设备，块特殊文件对应磁盘等执行面向块 I/O 操作的设备。

5. /etc

/etc 目录主要存放各种配置文件，该目录下不包括任何二进制文件。该目录下的文件主要提供给系统管理员对系统进行配置，普通用户对该目录下的文件一般只有读的权限，没有修改的权限。/etc 目录下包含各种硬件的配置文件，如显卡的配置和网卡的配置等，还包含系统中服务的配置文件，如 secure.conf。对应系统的各种配置的修改，基本都是在该目录下进行。这个目录是系统中最重要的目录之一。

6. /home

/home 目录存放用户的主目录。默认情况下，在 Linux 系统中创建的普通用户的主目录存放在/home 目录下。

7. /lib

/lib 目录下存放了各种编程语言的函数库。常见的 Linux 系统包含了 C、C++和 FORTRAN 语言的函数库。在使用 Linux 进行软件开发时，开发者可以直接调用这些已经写好的函数库。目录/lib/modules 包含了可加载的内核模块。/lib 目录存放了系统中所有重要的库文件，其他的库文件大部分存放在/usr/lib 目录下。

8. /lost+found

/lost+found 目录存放所有和其他目录没有关系的文件。在使用 fsck 对文件系统进行检查时，fsck 会将找到的文件存放在该目录下。

9. /mnt

/mnt 目录主要用来临时加载文件系统。这个目录包含了光驱、硬盘和 U 盘的挂载点。当使用 mount 挂载了一个设备，例如光驱时，可以通过访问目录/mnt/cdrom 下的文件来访问相应光盘上的内容。

10. /opt

/opt 目录用来安装附加软件包。

11. /proc

/proc 目录包含了进程和系统的信息。例如，可以在/proc 里查看 CPU 的信息、内存的使用信息等。这个目录对于进行系统调优很重要。

12. sbin

/sbin、/usr/sbin 和/usr/local/sbin 目录下存放的可执行文件必须具有 root 权限的用户才能使用。如 shutdown、fdisk 和 fsck 等命令。

13. /tmp

/tmp 目录存放临时性的文件，一些命令和应用程序会用到这个目录，也可以使用这个目录存放自己的临时文件。例如 MySQL 数据库会在该目录下创建临时的 sock 文件。这个目录下的文件都被定时删除，以避免临时文件占满整个磁盘。

14. /usr

/usr 目录是 Linux 系统中最大的目录之一。它存放了可以在不同的主机间共享的只读数据。多数的 Linux 系统中，在/usr 目录下至少包含的目录及其内容见表 6.3。

15. /var

/var 目录存放容易发生改变的数据。系统运行中产生的大部分日志存放在/var/log 目录下。

表 6.3　/usr 目录下的子目录

子　目　录	内　　容
X11R6	第 11 版第 6 次发布的 X Window 系统以及相关文件
bin	大多数用户命令和解释程序，如 Perl 和 Python 等
doc	各种工具、应用软件和编程语言的文档
games	游戏和教育软件的执行程序
include	C/C++头文件和包含特殊头文件的目录，如 GNU C++头文件（/usr/include/g++）、系统特定的头文件（/usr/include/sys）等
lib	目标文件、编程语言和内部的二进制文件（不能被用户和 Shell 脚本直接执行）
local	系统管理员本地安装的软件，默认情况下该目录没有任何内容
man	Linux 命令、应用软件和工具的使用手册
sbin	系统管理员和系统守护进程使用的命令和工具
share	与体系结构无关的只读数据
src	Linux 和软件包管理软件（如 RPM）的源代码
tmp	/tmp 目录的符号连接

本章重点介绍了文件的权限管理内容，这部分内容仅是文件系统管理的基础内容，后面还会深入介绍其他管理性操作。由于涉及权限的配置，使得这部分内容变得比较烦琐，需要读者理清思路，认真掌握。

本章最后介绍了 FHS，对该内容是需要一个逐渐熟悉的过程的，不要求读者能全部记住每个目录的作用，仅要求了解并在使用时能适当地查阅该内容，这样就能逐渐地掌握该部分内容了。

第7章 Shell 基础

Shell 是 Linux 操作系统中非常重要的一个应用环境，具有高效灵活的特点。也正因如此，使得其表现出了繁复且多样的特点。本章将介绍 Shell 的基本概念与 bash 的基本功能。充分理解和掌握本章的内容，有助于提高对 Linux 命令行环境的驾驭能力，也会为后续课程打下一个良好的基础。

7.1 认识 Shell

7.1.1 什么是 Shell

Shell 用最简单的叙述可以解释为是用户与操作系统内核之间的一个界面。在 Linux 系统中，实际调用硬件功能完成处理任务的是内核，而内核仅支持特定代码（二进制代码）的工作请求。系统中的应用程序是通过 API（Application Programming Interface，应用程序编程接口）与内核进行交互的，而普通用户则是通过 Shell 与内核进行交互的。

Shell 负责将用户命令解释为内核代码交由内核执行，同时将内核执行的结果解释为用户可识别的信息返回给用户。可见，Shell 在这里充当的是一个命令解释器的角色。

作为一种命令解释器，Shell 的表现形式是多样的，但从根本上可以分为图形界面和命令行界面两种。本书前面介绍的 Gnome 和 Kde 都是图形界面的命令解释器，而在终端环境下使用的都是命令行界面的 Shell。这两种 Shell 主要的区别就在于易用性和效率两个方面。图形界面的操作不是本书的重点，此处仅讨论命令行界面的 Shell，本书中以后所称的 Shell 若没有特别说明均指命令行界面下的 Shell。

Shell 除具有命令解释器的功能之外，还提供了一系列的结构化控制语句，从而使 Shell 具有程序设计语言的能力。利用该功能可以编写 Shell 脚本，实现对系统的自动化控制功能。

可见，Shell 的功能表现为两个方面。一方面是命令解释器功能，另一方面是编程语言的功能。本章将重点介绍 Shell 的命令解释器功能，有关 Shell 脚本的功能将在后续的专门章节介绍。

7.1.2 系统内的标准 Shell

Shell 的本质还是一种应用程序，由于 Linux 对 Shell 应用程序采用的是自由开放的策略，所以在 Linux 下有多种 Shell 应用程序可供用户选择。但大多数 Linux 发行版默认

采用的多是 bash 这种 Shell。因此，本章以 Bash 为主介绍 Shell 的各项功能。

在 UNIX 第一版出现之后，为了使用户能与操作系统有一个良好的沟通接口，AT&T 的 S.R.Bourne 开发了第一个 Shell，称为 Bourne Shell，简称 sh。这个 Shell 得到了普遍的应用，成为后续 Shell 发展的一个基础。

除 sh 之外，由美国加州大学伯克利分校（Berkeley）开发的 BSD C Shell（简称 csh）是另外一个重要的 Shell 分支。它有着类似 C 语言的语法规范，以及良好的交互能力。

在 sh 和 csh 的基础上发展出了众多的 Shell 分支。

（1）ash：即 Small Bourne Shell。由 sh 发展而来，是 Linux 中最小的 Shell，内置的命令也最少。其可执行文件是/bin/ash。

（2）ksh：即 Korn Shell，在 Linux 系统中是 Public Domain Korn Shell，也是由 sh 发展而来的。ksh 的一个不足是缺乏工作控制功能，其可执行文件是/bin/ksh。

（3）tcsh：即 Enhanced C Shell，是 csh 的增强版。该版本强化了 Shell 的用户交互能力，其可执行文件为/bin/tcsh。

（4）bash：即 Bourne Again Shell，是 sh 的增强版本，完全兼容 sh。bash 既具有 csh 的优秀的程序设计能力，又保持了良好的用户交互性，是当前各 Linux 发行版所默认采用的 Shell。

（5）zsh：是 Linux 中最大的 Shell 之一，内置了 80 多个命令，功能十分强大。其可执行文件为/bin/zsh。

以上所介绍的 Shell 被大多数 Linux 发行版所默认支持。Linux 系统中支持的可供用户选择使用的 Shell 被记录在/etc/shells 文件中，该文件所记录的内容会作为其他应用程序判断 Shell 存在的依据，该文件的内容如下：

```
[root@xinya ~]# cat /etc/shells
/bin/sh
/bin/bash
/sbin/nologin
/bin/tcsh
/bin/csh
/bin/ksh
```

当用户登录系统后，系统会自动为用户分配和启动一个 Shell，以作为用户与操作系统内核之间的接口，这个 Shell 称为“登录 Shell”。登录 Shell 是管理员在创建用户时定义的，可以使用“echo $SHELL”这个命令来查看当前用户的登录 Shell。

```
[root@xinya ~]# echo $SHELL
/bin/bash
```

7.1.3 bash 的功能

bash 的功能是多种多样的，可将其功能概括为以下 11 个方面。

（1）内置命令功能。

（2）自动补全功能。
（3）别名功能。
（4）历史命令功能。
（5）通配符功能。
（6）重定向功能。
（7）管道功能。
（8）计算功能。
（9）指令替代功能。
（10）子 Shell 功能。
（11）环境配置功能。

本章的后续内容将以上述 Shell 的功能为主线，依次详细介绍各项功能。

7.2　bash 的基本功能

本节将介绍 bash 的内置命令功能、自动补全功能、别名功能、历史命令功能和通配符功能。

7.2.1　bash 的内置命令功能

在 Linux 系统中，命令分为内部命令和外部命令两种。

外部命令是以可执行文件的形式存在于文件系统中供用户调用和执行的命令。

内部命令是指命令是由 Shell 本身提供的，不存在可执行文件的形式。

bash 提供了一系列内部命令，如 cd、pwd 和 test 等。Linux 提供了 type 命令供用户查看一个命令是属于内部命令还是外部命令。

```
[root@xinya ~]# type pwd
pwd is a shell builtin  ←表示 pwd 命令是内部命令
[root@xinya ~]# type mkdir
mkdir is /bin/mkdir  ←表示 mkdir 命令是外部命令，其可执行文件为/bin/mkdir
```

下面对 type 命令的用法加以说明。

type 命令用于显示命令的来源。

【语法】**type　[选项]　命令名**

选项说明如下：

-t：以 file 表示外部命令、以 alias 表示别名命令、以 builtin 表示内部命令的方式显示命令的来源。

-p：显示外部命令的完整路径。

-a：显示 PATH 路径中所有匹配命令名的命令。

【示例】显示命令的来源信息。

```
[root@xinya ~]# type -t pwd
```

```
builtin  ← pwd 是内部命令
[root@xinya ~]# type -t rm
alias  ←rm 是一个别名命令
[root@xinya ~]# type -t man
file  ←man 是一个外部命令
```

7.2.2 bash 的自动补全功能

自动补全是 bash 为方便键盘输入而提供的一项功能。该功能可以用于自动补全文件名和命令名。

bash 的自动补全功能允许用户使用 Tab 键自动补全待输入的命令和文件名。

```
[root@xinya ~]# ls
Desktop  file1  file2  mbox  script  xorg.conf.bak
[root@xinya ~]# cat xor【Tab】g.conf.bak
                          ↑ 此处按 Tab 键会自动补全剩余的文件名
[root@xinya ~]# shu【Tab】tdown
                     ↑此处按 Tab 键会自动补全剩余的命令名
```

注意，Tab 键的自动补全功能实际上是基于当前命令集合或文件集合来实现的。在利用 Tab 键补全文件名的例子中，当前的文件集合就是当前目录下的所有文件，包括 Desktop、file1、file2、mbox、script 和 xorg.conf.bak，在这个文件集合中以“xor”开头的文件只有一个，即 xorg.conf.bak，所以在 xor 后直接使用 Tab 键即可自动补全该文件名的后续字符。事实上，在当前文件集合中，只有一个文件名以“x”开头，所以在“x”字符后使用 Tab 键同样可以补全剩余的文件名。

```
[root@xinya ~]# ls
Desktop  file1  file2  mbox  script  xorg.conf.bak
[root@xinya ~]# cat x【Tab】org.conf.bak
                        ↑ 此处按 Tab 键会自动补全剩余的文件名
```

对于命令的自动补全功能也是一样，Linux 会在系统的命令集合中筛选以特定字符开头的命令，并自动补全后续字符。在上面的 shutdown 命令的自动补全操作中，当前的命令集合中只有一个命令 shutdown 以“shu”开头，所以 Tab 会自动补全剩余的命令拼写。在空白命令行中使用两次 Tab 键，会显示当前系统的命令集合。

```
[root@xinya ~]#【Tab】【Tab】                ←连续使用两次 Tab 键
Display all 2883 possibilities? (y or n)   ←是否显示 2883 条命令,回答"y"
...
showkey
showmount
showrgb
shred
shutdown                     ←在命令集合中,只有该命令以"shu"字符串开头
signtool
```

```
signver
sim_client
sim_server
...
```

利用 Tab 键实现的自动补全功能可提高对命令和文件名的输入效率。在使用这个功能时还要注意，当连按两次 Tab 键时，bash 会将所有匹配的可能显示出来。

```
[root@xinya ~]# ls
Desktop  file1  file2  mbox  script  xorg.conf.bak
[root@xinya ~]# cat file【Tab】【Tab】     ←此处连按两次 Tab 键
file1  file2                              ←bash 会将所有可能的匹配显示出来
[root@xinya ~]# sho 【Tab】【Tab】         ←此处连按两次 Tab 键
shopt        showconsolefont  showmount   ←bash 会显示所有以 sho 开头的命令
showchar     showkey          showrgb
```

7.2.3 bash 的命令别名功能

别名是 bash 提供的又一项重要的功能。该功能允许为一个命令定义一个别名，利用别名可以调用该别名所代表的命令操作。

当前系统中默认存在一系列的命令别名，可以使用 alias 命令来具体查看当前系统的别名列表。

```
[root@xinya ~]# alias
alias cp='cp -i'
        ←cp 命令是"cp -i"命令的别名,-i 选项的作用是覆盖目标文件前先询问用户
alias l.='ls -d .* --color=tty'
alias ll='ls -l --color=tty'
alias ls='ls --color=tty'
alias mv='mv -i'
alias rm='rm -i'
        ←rm 命令是"rm  -i"命令的别名,-i 选项的作用是删除目标文件前先询问用户
alias which='alias | /usr/bin/which --tty-only --read-alias --show-dot
--show-tilde'
```

除系统提供的别名命令之外，用户也可以利用 alias 命令来自定义别名命令。其语法结构为：

alias 别名='命令描述'

注意，命令描述最好使用单引号“'”括起来，以避免空格和特殊字符导致的定义不完整。

【示例】定义别名命令 dir，用于显示目录下的文件和子目录的详细信息，并且以易读的方式显示文件和目录的容量信息。

```
[root@xinya ~]# alias dir='ls -lh --color=tty'
```

```
[root@xinya ~]# alias
alias cp='cp -i'
alias dir='ls -lh --color=tty'
alias l.='ls -d .* --color=tty'
alias ll='ls -l --color=tty'
alias ls='ls --color=tty'
alias mv='mv -i'
alias rm='rm -i'
alias which='alias | /usr/bin/which --tty-only --read-alias --show-dot
--show-tilde'
[root@xinya ~]# dir
total 32K
drwxr-xr-x 3 root root 4.0K  12-31 21:01 Desktop
-rwxr--r-- 1 root root    4  01-18 05:01 file1
-rw-r--r-- 1 root root    0  01-18 05:09 file2
-rw------- 1 root root  16K  01-20 17:30 mbox
drwxr-xr-x 2 root root 4.0K  01-19 00:22 script
```

利用别名命令可以简化命令行的复杂程度，强化对特定命令参数的使用强度，以及自定义命令的拼写方法，以适应不同的环境需要。

需要注意的一点是，当在命令行环境中输入一个命令后，bash 首先会判断该命令是否是内部命令，如果是则直接执行，如果不是，bash 会判断命令是否为别名命令，如果是别名命令则执行别名命令后的指令描述，如果不是才会查找文件系统中的可执行文件执行。所以，若不需要使用别名时，可以直接在命令行中指定待执行命令的完整路径信息。

```
[root@xinya ~]# ls
Desktop  file1  file2  mbox  script  xorg.conf.bak   ←文件名有颜色信息
[root@xinya ~]# /bin/ls
Desktop  file1  file2  mbox  script  xorg.conf.bak   ←文件名没有颜色信息
```

别名命令的取消可以使用 unalias 命令。

unalias　别名

【示例】取消 dir 别名命令。

```
[root@xinya ~]# alias
alias cp='cp -i'
alias dir='ls -lh --color=tty'
alias l.='ls -d .* --color=tty'
…
[root@xinya ~]# unalias dir
[root@xinya ~]# alias
alias cp='cp -i'
alias l.='ls -d .* --color=tty'
…
```

上述对命令别名的定义仅在本次登录期间有效，当该用户退出登录后所定义的别名命令将归于无效，若需要保留所定义的别名，需要在后续内容中提到的环境变量配置文件中进行定义。

下面对 alias 命令的用法加以说明。

alias 命令是 bash 的一个内置命令，用于显示和定义命令的别名。

alias	显示系统中的别名命令。
alias 别名='命令描述'	定义别名命令。
unalias 别名	删除别名定义。

7.2.4 bash 的历史命令功能

历史命令功能是指 bash 会记录、存储和允许用户调用曾经输入过的命令。该功能可跟踪用户最近的操作，并允许调用以前的命令操作。

查看当前登录的历史命令可以使用 history 命令。

```
[root@xinya ~]# history
   41  vim file2 file3
   42  cleawr
   43  clear
   44  vim file2
   45  vim file2
   46  vim file2
   47  ls -a
   48  vi .viminfo
…
 1032  alias
 1033  man alias
 1034  man 8 alias
 1035  man 1 alias
 1036  type
 1037  type alias\
 1038  clear
 1039  history
```

history 命令的输出格式为“历史命令编号　命令”。history 命令还可以使用一个 *n*（*n* 为数字）的选项，用于定义显示最近使用的 *n* 条命令。如查看最近使用的 10 条命令，可以使用“history 10”命令。

```
[root@xinya ~]# history 10 ←查看最近使用的 10 条命令
 1038  clear
 1039  history
 1040  history | more
```

```
1041  clear
1042  history -n 10
1043  history -n 5
1044  man history
1045  history  5
1046  history
1047  history 10
```

在得到历史命令列表后如何调用历史命令呢？调用历史命令的方法主要有以下 4 种。

（1）使用↑和↓方向键调用历史命令。方向键↑可以向上逐一显示已输入的历史命令，方向键↓可以向下逐一显示已输入的历史命令。当选定特定的历史命令后，直接按回车键开始执行。

（2）使用“！历史命令编号”来调用特定的历史命令。通过 history 命令可得到历史命令的编号，使用“！历史命令编号”可直接调用特定的历史命令。

```
[root@xinya ~]# history 10
 1041  ls
 1042  touch a b c
 1043  tar -cvf ac.tar
 1044  tar -cvf ac.tar a c
 1045  tar -cvf bc.tar b c
 1046  tar -tf ac.tar bc.tar
 1047  tar -tf ac.tar c
 1048  tar -df ac.tar c
 1049  history
 1050  history 10
[root@xinya ~]# !1041                      ←执行第 1041 号历史命令
ls
Desktop  file1  file2  mbox  script  xorg.conf.bak
```

（3）使用“！字符串”来调用历史命令中最近输入的，以给出的字符串开头的历史命令。

（4）使用“!!”来调用最近输入的一条历史命令，即上一条历史命令。

在了解了历史命令的查看和调用方法后，还要认识历史命令的实现机制，即系统如何记录提供历史操作记录的。大体上说，历史命令的形成机制是这样的。

当用户在命令行输入命令并执行后，该命令会记录到系统缓存中，当用户从系统注销时，位于缓存中的历史命令会被写入~/.bash_history 文件中，这样，历史命令就被以文件的形式保存下来了。当用户再次登录系统时，bash 会由~/.bash_history 文件中读取历史命令记录并加载至系统缓存中，作为历史命令供用户调用。

系统默认历史命令的容量为 1000 条，当输入的命令超过 1000 条时，新输入的命令会将最早的命令挤出~/.bash_history 文件，所以当使用 history 命令读取~/.bash_history 文件的缓存镜像时第一条历史命令的编号不一定就是 1。

在 bash 的历史命令实现机制中，~/.bash_history 文件是非常重要的一个环节，用户

的操作会在用户退出登录时保存在该文件中。但是，当一个用户以多种方式（如在多个终端下）同时登录系统时，究竟记录用户在哪个终端下进行的操作呢？因为历史命令的记录是在用户退出登录后记录到~/.bash_history 文件中的，bash 认为最后一个退出登录的用户才意味着该用户完全退出系统，所以 bash 仅会将最后一个注销登录的用户所做的操作记录至~/.bash_history 文件作为该用户的历史命令使用。

对于 bash 的历史命令功能而言，主要是通过 history 命令来查看、控制和操作的。

下面对 history 命令的用法加以说明。

【语法】**history 【选项】 【文件名】**

选项说明如下：

当不使用任何选项时，默认显示当前 Shell 缓存中的所有历史命令。

n ：*n* 为数字，用于显示最近输入的 *n* 条历史命令。

-c ：将当前 Shell 缓存中的历史命令清空。

-a [文件名]：将当前缓存中新增的历史命令写入指定的文件中。若未指定文件，则写入~/.bash_history 文件中。

-r [文件名]：将指定文件中的历史命令读入到当前缓存中。若未指定文件，则默认读取~/.bash_history 文件中的历史命令。

-w[文件名]：将当前 Shell 缓存中的所有历史命令写入指定的文件中，若未指定文件，则写入~/.bash_history 文件中。

尽管历史命令能够追踪用户过去的操作，但是一个局限是历史命令功能不能记录命令的操作时间。

7.2.5 bash 的通配符功能

通配符是 Shell 中的一种非常重要的匹配字符，用于匹配一个或多个任意字符，在文件名的查找、修改和删除等相关操作中，通配符有着十分广泛的用途。

bash 提供了 3 种通配符供用户使用，分别是*、？和[ ]。

（1）*：代表 0 到任意多个任意字符。下面通过示例来了解该通配符。

```
[root@xinya ~]# touch 123abc defabc tyuabc abc ruyabc
#创建了 5 个文件名中包含有 abc 字符串的文件
[root@xinya ~]# ls -t
123abc  abc  defabc  ruyabc  tyuabc  xorg.conf.bak  mbox  script  file2
file1  Desktop
[root@xinya ~]# ls *abc
123abc  abc  defabc  ruyabc  tyuabc
```

上例中，由于*号表示 0 到任意多个任意字符，所以“ls *abc”命令的输出不仅包括各个含有 abc 字符串的文件名，还包括“abc”这个文件名。

（2）？：代表 1 个任意字符。

```
[root@xinya ~]# ls
Desktop  file1  file2  mbox  script  xorg.conf.bak
```

```
[root@xinya ~]# ls file?
file1  file2
```

此处“？”号表示 1 个任意字符，因此会得到 file1 和 file2。

（3）[ ]：包含匹配，[ ]内可以包含字符或字符范围等信息，表示包含这些字符或字符范围。

【示例】查询/etc/init.d/目录下以 g、t 或 v 开头的文件名。

```
[root@xinya ~]# ls /etc/init.d/[gtv]*
/etc/init.d/gpm  /etc/init.d/tcsd  /etc/init.d/vncserver
```

注意，包含匹配是匹配各个字符之一而不是匹配整个字符串。

包含匹配可以用于表示字符和字符范围。其写法为：

[abcdef]：表示包含 a，或包含 b、或包含 c，或包含 d，或包含 e，或包含 f，而不是包含 abcdef。

[a-z]：表示的是一个字符范围，包含 a，或包含 b，或包含 c，……或包含 z 的任意小写字母。

[0-9]：表示的是一个数字范围，包含 0～9 之间的任意一个数字。

[a-zA-Z0-9]：表示所有小写字母、大写字母和所有数字。

[^字符串]：非包含匹配，“[^…]”中括号内可以包含字符或字符范围等信息，表示不包含这些字符或字符范围。它与[ ]包含匹配的作用正好相反。

【示例】查询/etc/init.d/目录下不是以 g、t 或 v 开头的文件名。

```
 [root@xinya ~]# ls /etc/init.d/[^gtv]*
/etc/init.d/NetworkManager  /etc/init.d/ibmasm     /etc/init.d/oddjobd
/etc/init.d/acpid           /etc/init.d/ip6tables  /etc/init.d/pand
/etc/init.d/anacron         /etc/init.d/iptables   /etc/init.d/pcscd
…
```

注意，非包含匹配的表示方法与包含匹配相同，此处就不再赘述了。

7.3 输入/输出重定向功能

输入重定向和输出重定向是 bash 提供的又一项重要的功能，利用该功能可方便地对进程的标准输入和标准输出进行修改以获得更符合用户需求的程序运行方式。

在 Linux 操作系统中，一个程序被加载至内存并开始运行后是以进程的形态表现出来的，而系统中的每一个进程在默认的情况下都会有 3 个标准的通道，称为标准输入（stdin）、标准输出（stdout）和标准错误输出（stderr），如图 7.1 所示。

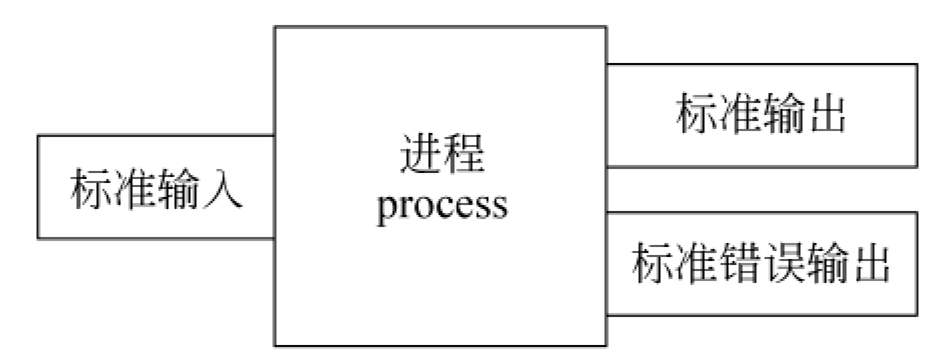


图 7.1　进程的标准通道

（1）标准输入（stdin）：是进程所需处理的数据的默认来源，系统中进程的标准输入默认是

键盘或指定的参数文件。标准输入的操作代码为 0。

（2）标准输出（stdout）：是进程将处理结果输出的默认位置，系统中进程的标准输出默认是屏幕。标准输出的操作代码为 1。

（3）标准错误输出（stderr）：是进程出现错误后，错误信息输出的默认位置，系统中进程的标准错误输出默认是屏幕。标准错误输出的操作代码为 2。

7.3.1 输入重定向

输入重定向是指将进程的标准输入由默认的键盘重定向（修改）为其他设备或文件。输入重定向使用“<”符号来表示。

下面以 cat 命令为例来说明输入重定向。

当在命令行单独使用 cat 命令而不添加任何参数时，cat 命令的作用是将标准输入（键盘）传递的字符打印（输出）到标准输出设备（屏幕）上。

```
[root@xinya ~]# cat
hello    ←在标准输入（键盘）设备上输入的内容
hello    ←被 cat 命令输出到标准输出设备（屏幕）上
welcome to linux  ←在标准输入（键盘）设备上输入的内容
welcome to linux  ←被 cat 命令输出到标准输出设备（屏幕）上
```

可以使用输入重定向来将 cat 命令的标准输入由键盘设备修改为指定的文件，则 cat 命令会将文件中的内容直接打印（显示）到标准输出设备（屏幕）上。

```
[root@xinya ~]# cat < file2
hello
welcome to Linux
```

file2 文件中的内容为“hello”和“welcome to Linux”。此处使用“<”符号将 cat 命令的标准输入设备由键盘重定向（修改）为文件 file2，cat 命令将会将文件 file2 中的所有内容显示到标准输出设备（屏幕）上。

尽管示例中“cat < file2”命令与“cat file2”命令的执行结果一样，但这两个命令的工作机制是不同的，“cat file2”是将 file2 文件作为一个参数使用，而“cat < file2”却是将 file2 作为一个标准输入设备使用的。

7.3.2 输出重定向

输出重定向是指将进程的标准输出由屏幕重定向至其他的文件或设备。输出重定向使用“>”符号表示。

下面以 ls 命令为例来说明输出重定向。

“ls -l”命令单独使用时的作用是在屏幕上显示当前目录下的文件和子目录。

```
[root@xinya ~]# ls -l
total 36
drwxr-xr-x  3 root root  4096   Dec 31  21:01 Desktop
```

```
-rwxr--r--  1 root root     4   Jan 18  05:01 file1
-rw-r--r--  1 root root    23   Feb  1  08:18 file2
-rw-------  1 root root 15527   Jan 20  17:30 mbox
drwxr-xr-x  2 root root  4096   Jan 19  00:22 script
-rw-r--r--  1 root root   568   Jan 21  12:50 xorg.conf.bak
```

在该命令中，“ls -l”命令将目录和文件信息输出至标准输出设备（屏幕）上。可以使用“>”符号来将“ls -l”命令的输出结果重定向至特定的文件中。

```
[root@xinya ~]# ls -l > dirlist
[root@xinya ~]# cat dirlist
total 36
drwxr-xr-x 3 root root  4096   Dec 31  21:01 Desktop
-rw-r--r-- 1 root root     0   Feb  1  08:47 dirlist
-rwxr--r-- 1 root root     4   Jan 18  05:01 file1
-rw-r--r-- 1 root root    23   Feb  1  08:18 file2
-rw------- 1 root root 15527   Jan 20  17:30 mbox
drwxr-xr-x 2 root root  4096   Jan 19  00:22 script
-rw-r--r-- 1 root root   568   Jan 21  12:50 xorg.conf.bak
```

上例中，“ls -l”命令的输出结果并未直接显示在屏幕上，而是被“>”输出重定向至 dirlist 文件中，通过“cat dirlist”命令可以知道文件 dirlist 中记录的是“ls -l”命令的输出结果。

可见，“>”输出重定向是将“ls -l”进程的标准输出由屏幕修改为文件，这就是输出重定向的作用。需要注意的是，输出重定向是一种覆盖式的操作。还是 dirlist 这个文件，文件中已包括了“ls -l”的输出结果了，这时再向这个文件中重定向内容时，新内容会覆盖文件中已有的内容。

```
[root@xinya ~]# cat dirlist
total 36
drwxr-xr-x 3 root root  4096 Dec 31 21:01 Desktop
-rw-r--r-- 1 root root     0 Feb  1 08:47 dirlist
-rwxr--r-- 1 root root     4 Jan 18 05:01 file1
-rw-r--r-- 1 root root    23 Feb  1 08:18 file2
-rw------- 1 root root 15527 Jan 20 17:30 mbox
drwxr-xr-x 2 root root  4096 Jan 19 00:22 script
-rw-r--r-- 1 root root   568 Jan 21 12:50 xorg.conf.bak
[root@xinya ~]# echo "stdout is a file" > dirlist
                           ←echo 命令的执行结果重定向至 dirlist 文件
[root@xinya ~]# cat dirlist
stdout is a file           ←新进入文件的内容覆盖了文件已有的内容
```

上例使用的是 echo 回显命令，该命令默认会在屏幕上打印后续的字符串，上例将该命令的标准输出由屏幕重定向为文件 dirlist。由示例中可见，echo 命令的执行结果确实将内容重定向输出至 dirlist 文件，但该输出覆盖了 dirlist 文件中原有的内容。

在使用输出重定向时，除使用“>”这种覆盖式重定向外，还可使用“>>”这种追加式重定向，即新内容会追加到已有内容的尾部而不是覆盖已有的内容，这种特性有利于在日志操作中连续记录操作内容。

```
[root@xinya ~]# cat dirlist
stdout is a file            ←文件中已有的内容
[root@xinya ~]# echo "second output" >> dirlist
                            ←向文件 dirlist 中追加重定向
[root@xinya ~]# echo "third output" >> dirlist
                            ←向文件 dirlist 中追加重定向
[root@xinya ~]# cat dirlist
stdout is a file
second output               ←追加至文件尾部的新内容
third output                ←追加至文件尾部的新内容
```

7.3.3 错误输出重定向

错误输出重定向是指将进程的标准错误输出由屏幕重定向至其他设备或文件，错误输出重定向使用“2>”或“2>>”表示。“2>”是覆盖式的错误输出重定向，“2>>”是追加式的错误输出重定向。

下面还是以“ls -l”命令为例来说明错误输出重定向。

```
[root@xinya ~]# ls -l /kde
ls: /kde: No such file or directory    ←由于没有/kde 目录,所以出现错误提示
[root@xinya ~]# ls -l /kde > dirlist
ls: /kde: No such file or directory
```

上例中，由于没有/kde 这个目录，所以“ls -l /kde”命令会提示一个错误“ls: /kde: No such file or directory”。注意，这是一个错误输出而不是一个标准输出，所以尽管使用了“ls -l /kde > dirlist”命令，仍然会出现“ls: /kde: No such file or directory”这个错误提示。

对于这种进程的标准错误输出可以使用“2>”或“2>>”实现标准错误输出重定向。

```
[root@xinya ~]# ls -l /kde 2> errfile  ←将 ls -l /kde 进程的错误输出重定向
[root@xinya ~]# cat errfile
ls: /kde: No such file or directory
```

需要注意的是“2>”是一种覆盖式的错误输出重定向；若需要在一个文件中连续记录错误内容，可以使用“2>>”这种覆盖式的错误输出重定向。

在输出重定向中，若需要将进程的标准输出和标准错误输出同时定位至一个文件，可以联合使用输出重定向和错误输出重定向。

```
[root@xinya ~]# ls script yum
ls: yum: No such file or directory ←由于没有 yum 目录,所以会出现错误提示
script:                            ←script 目录的输出结果
sh01.sh  sh03.sh  sh05.sh  sh07.sh  sh09.sh  sh11.sh
```

```
sh02.sh  sh04.sh  sh06.sh  sh08.sh  sh10.sh
[root@xinya ~]# ls script yum > outfile 2>&1
                    ←将命令的标准输出和标准错误输出均重定向至 outfile 文件
[root@xinya ~]# ls script yum &> outfile
                    ←将命令的标准输出和标准错误输出均重定向至 outfile 文件
[root@xinya ~]# cat outfile
ls: yum: No such file or directory
script:
sh01.sh
…
```

上例中将标准输出和标准错误输出重定向至同一文件时使用了两种语法结构，分别是“> 文件名 2>&1”和“&>”。这两种语法结构都可以实现将标准输出和标准错误输出统一重定向至特定文件的目的。

“ls script yum>outfile 2>&1”的语法含义是将“ls script yum”命令的标准输出重定向至 outfile 文件，“ 2>&1”的含义是错误输出“2>”与标准输出相同“&1”（标准输出的操作代码为 1）。

“ls script yum &>outfile”的语法含义是将“ls script yum”命令的标准输出和标准错误输出均重定向至 outfile 文件，“&>”中的&表示标准输出 1 和标准错误输出 2。

在输入/输出重定向这个主题的最后，还需要向大家介绍两个常用的文件：/dev/null 和/dev/zero，这两个文件因具有特殊的含义，所以在输入/输出重定向处理中被经常使用。

/dev/null 和/dev/zero 这两个文件是一种数据槽文件，其特点如下。

（1）当数据写入这两个文件中时，写入的数据将全部被丢弃掉，相当于一个数据垃圾桶。

（2）当由/dev/null 文件读取数据时，该文件总会返回文件结束的信息。

（3）当由/dev/zero 文件读取数据时，该文件总会返回“\0”字符。

当需要将进程的标准输出或标准错误输出丢弃时，可以利用这两个文件来实现。

```
[root@xinya ~]# ls script yum > outfile 2>/dev/null
        ←将进程的标准输出重定向至 outfile 文件,将错误输出重定向至/dev/null 文件
[root@xinya ~]# cat outfile          ←文件中只有"ls script yum"的标准输出
script:
sh01.sh
sh02.sh
sh03.sh
sh04.sh
…
```

7.4 bash 的管道功能

与输入/输出重定向类似，bash 的管道功能也是针对进程的标准输入/输出通道所做的操作。与输入/输出重定向不同的是，管道功能的主要作用是对不同进程的标准输入和

标准输出进行连接。

如图 7.2 所示，管道的作用是将“进程 1”的标准输出作为“进程 2”的标准输入交给“进程 2”来使用。这里，管道实际上起到的是连接两个进程的作用。

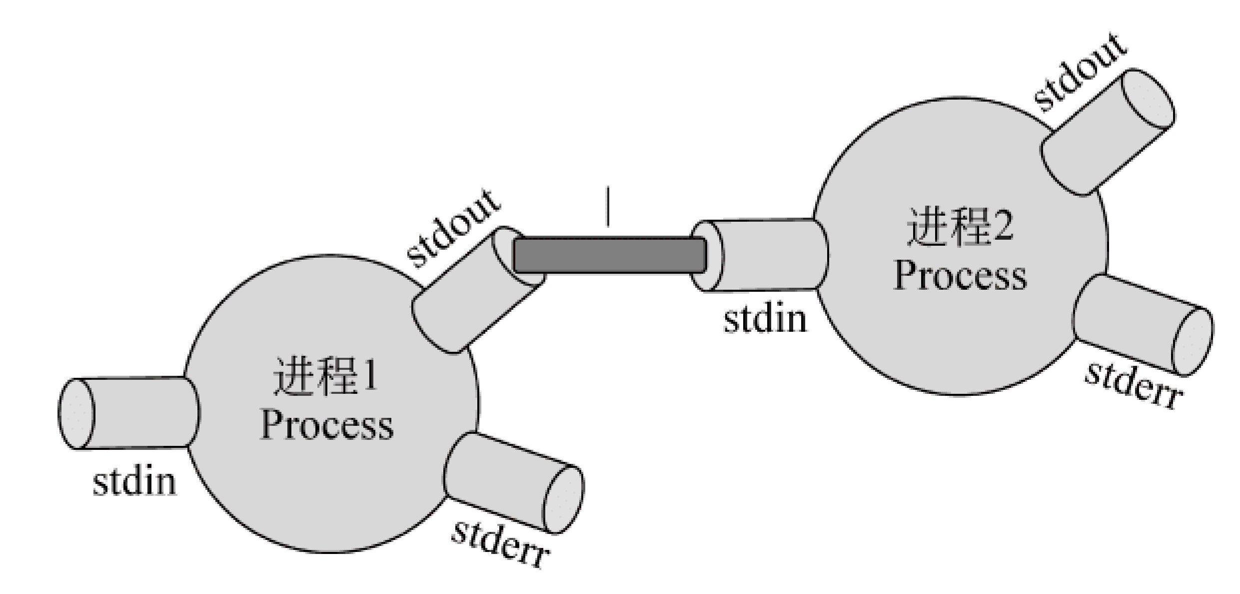


图 7.2　用于连接两个进程的管道

7.4.1　管道命令的使用方法

在 bash 中使用“|”来表示管道，其语法结构为“命令 A|命令 B”，即将“命令 A”的标准输出作为“命令 B”的标准输入使用。

```
[root@xinya ~]# ls -l /etc | more
total 3800
-rw-r--r--  1 root root   2518     Mar  1  2010 DIR_COLORS
-rw-r--r--  1 root root   2420     Mar  1  2010 DIR_COLORS.xterm
-rw-r--r--  1 root root  92794     Jun  4  2007 Muttrc
-rw-r--r--  1 root root      0     Jun  4  2007 Muttrc.local
drwxr-xr-x  4 root root   4096     Mar 14  2011 NetworkManager
drwxr-xr-x  8 root root   4096     Jan 21  13:05 X11
…
--More--
```

在上例中，“ls -l /etc”会得到许多内容，终端会全部显示该命令的输出结果，以至于一屏无法显示而只能显示命令输出最后一屏的内容。当需要查看前面的内容时在终端环境下就显得无能为力了。本例中利用了 more 命令的分屏显示的功能，将“ls -l /etc”命令的输出结果交给默认命令分屏显示，实现了对“ls -l /etc”命令的输出结果分屏查看的目的。

管道命令在管理工作中是被经常使用的，在使用管道的过程需要注意两个问题。

（1）管道仅处理标准输出，而不处理标准错误输出。

（2）管道后的命令必须能接收来自前一个命令的输出数据作为自己的标准输入使用。

在管道的应用中，有一些很常用的命令需要特别注意，利用这些命令可以很方便地实现相关的管理性操作。下面将重点介绍几个常用的管道命令。

7.4.2　数据选取命令 cut

数据选取命令 cut 主要用于选取文件或标准输入中特定字段的数据。该命令是以行

为单位进行操作的。

【示例】选取/etc/passwd 文件中用户名和用户 UID 两个字段的内容。

/etc/passwd 文件主要用于记录用户信息，每行是一条用户信息。行中使用“：”号将用户信息分为 7 个字段，用户名处于第一个字段，UID 处于第 3 个字段。因此根据题目，要以“：”为分隔符选取每行的第一个和第三个字段的内容。

```
[root@xinya ~]# cut  -d :  -f 1,3  /etc/passwd
root:0
bin:1
daemon:2
adm:3
lp:4
…
```

注意，示例中的 cut 命令使用-d 选项来定义行中各个字段间的分隔符为“：”号（-d ：），使用“-f”选项来定义要选取第 1 字段和第 3 字段（-f 1,3）。

下面对 cut 命令的用法加以说明。

【语法】**cut [选项] 参数**

选项说明如下：

-d 分隔符：用于定义行中字段间的分隔符。

-f：用于定义待选取的字段，字段的写法为 n（表示选取第 n 个字段，n 为数字）、n_1，n_2，…（表示选取 n_1、n_2 等不连续的字段）或 n_1-n_x（表示选取从 n_1 开始到 n_x 结束的连续字段）。

-c 长度：以字符为单位，定义选取字符的长度。

cut 命令在管道中使用。

【示例】使用 cut 命令选取 ifconfig 命令输出的本机 IP 地址。

```
[root@xinya ~]# ifconfig | head -n 2 | tail -n 1 | cut -d : -f 2 | cut -d
' ' -f 1
192.168.1.109
```

7.4.3 数据过滤命令 grep

grep 命令是一个以行为单位的数据过滤命令，允许用户定义字符串，并过滤包含用户定义字符串的行。

【示例】查找/etc/passwd 文件中包含 root 字符串的行。

```
[root@xinya ~]# grep "root"  /etc/passwd
root:x:0:0:root:/root:/bin/bash
operator:x:11:0:operator:/root:/sbin/nologin
```

下面对 grep 命令的用法加以说明。

【语法】**grep "字符串" 文件名**

grep [选项] "字符串" 文件名

选项说明如下：

-a：对二进制文件以文本文件的方式查找数据。

-c：计算找到的行数。

-i：忽略大小写的不同。

-n：输出匹配的行的同时输出该行的行号。

-v：反向选择。

--color=auto：为匹配的字符串添加颜色。

【示例】查找/etc/passwd 文件中以/bin/bash 作为登录 Shell 的用户信息

```
[root@xinya ~]# grep "/bin/bash" /etc/passwd
root:x:0:0:root:/root:/bin/bash
user1:x:500:500::/home/user1:/bin/bash
user2:x:501:501::/home/user2:/bin/bash
…
```

【示例】查找/etc/passwd 文件中以/bin/bash 作为登录 Shell 的用户信息，并用颜色标识“/bin/bash”字符串。

```
[root@xinya ~]# grep --color=auto "/bin/bash" /etc/passwd
root:x:0:0:root:/root:/bin/bash
user1:x:500:500::/home/user1:/bin/bash
user2:x:501:501::/home/user2:/bin/bash
user3:x:502:502::/home/user3:/bin/bash
admin1:x:503:503::/home/admin1:/bin/bash
admin2:x:504:504::/home/admin2:/bin/bash
admin3:x:505:505::/home/admin3:/bin/bash
test:x:506:507::/home/test:/bin/bash
te:x:509:509:::/bin/bash
```

【示例】统计/etc/passwd 文件中有多少用户以/bin/bash 作为登录 Shell。

```
[root@xinya ~]# grep -c "/bin/bash" /etc/passwd
9
```

【示例】显示/etc/passwd 文件中不是以/bin/bash 作为登录 Shell 的用户信息。

```
[root@xinya ~]# grep -v "/bin/bash" /etc/passwd
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
```

```
news:x:9:13:news:/etc/news:
…
```

grep 命令也是管道中经常使用的命令。

【示例】利用 grep 命令过滤 ifconfig 命令的输出结果中包含 IP 信息的行。

在 ifconfig 命令的输出中，IP 地址信息均以“inet addr:”来定义。

```
[root@xinya ~]# ifconfig | grep "inet addr"
          inet addr:192.168.1.109  Bcast:255.255.255.255
          Mask:255.255.255.0
          inet addr:127.0.0.1  Mask:255.0.0.0
[root@xinya ~]# ifconfig | grep "inet addr" | cut -d : -f 2 | cut -d '
' -f 1
192.168.1.109
127.0.0.1
```

7.4.4 数据排序命令 sort

数据排序命令 sort 的作用在于以行为单位，按特定字段对数据进行排序。

【示例】查看/etc/passwd 文件中的用户名与 UID 信息。

```
[root@xinya ~]# cat /etc/passwd | cut -d : -f 1,3
root:0
bin:1
daemon:2
adm:3
sync:5
shutdown:6
halt:7
mail:8
news:9
uucp:10
operator:11
games:12
gopher:13
ftp:14
nobody:99 ←注意，UID 并非按顺序排列
nscd:28
vcsa:69
rpc:32
apache:48
…
```

【示例】查看/etc/passwd 文件中的用户名与 UID 信息，并按 UID 顺序对输出信息进行排序。

```
[root@xinya ~]# cat /etc/passwd | cut -d : -f 1,3 | sort -t : -k 2
root:0
bin:1
uucp:10
avahi-autoipd:100
operator:11
games:12
gopher:13
ftp:14
daemon:2
nscd:28
rpcuser:29
adm:3
rpc:32
ntp:38
```

此处使用了 sort 命令，该命令首先使用-t 选项定义字段的分隔符，然后使用-k 选项定义按哪个字段来进行排序。由输出可见，输出内容按第二个字段也就是 UID 进行了排序，但排序规则却并非是按数字顺序进行的，而是按 ASCII 码的顺序（字符顺序）进行的。

需要注意的是，sort 在默认的情况下对特定的字段使用 ASCII 码顺序（字符顺序）进行排序，当需要使用数字进行排序时需要使用-n 选项。

```
[root@xinya ~]# cat /etc/passwd | cut -d : -f 1,3 | sort -t : -k 2 -n
root:0
bin:1
daemon:2
adm:3
lp:4
sync:5
shutdown:6
halt:7
mail:8
news:9
uucp:10
…
```

下面对 sort 命令的用法加以说明。

【语法】**sort ［ 选项 ］**

选项说明如下：

-t 分隔符：用于定义字段间的分隔符。

-k 字段号：用户定义按哪个字段进行排序。

-n：使用数字进行排序。

-r：反向排序，默认的情况下 sort 命令按升序排序，反向排序实现的是降序排序。

-f：排序过程中忽略大小写的不同。

-M：使用月份进行排序。

7.4.5 重复内容过滤命令 uniq

重复内容过滤命令 uniq 用于对输入的内容进行重复性过滤，即若输入的数据中相邻行若存在重复，则仅保留其中一行。

【示例】查询/etc/passwd 文件中共有几种登录 Shell。

```
[root@xinya ~]# cut -d : -f 7 /etc/passwd ←显示/etc/passwd 文件中的登录 Shell
/bin/bash
/sbin/nologin
…
/sbin/nologin
/bin/sync
/sbin/shutdown
/sbin/halt
/sbin/nologin
…
/sbin/nologin
/bin/bash
…
/bin/bash
[root@xinya ~]# cut -d : -f 7 /etc/passwd | uniq
                                    ←对输出的相同内容的数据进行过滤
/bin/bash
/sbin/nologin
/bin/sync
/sbin/shutdown
/sbin/halt
/sbin/nologin
/sbin/nologin
/bin/bash
```

上例中，尽管使用了 uniq 对重复数据进行过滤，但是还是存在多个重复的数据，包括两个/bin/bash 和 3 个/sbin/nologin。产生这一问题的原因在于，uniq 只能对相邻行的重复内容进行过滤，若重复的内容处于不相邻的行，则 uniq 无法完成过滤，所以为了完整地过滤重复信息，在过滤前最好对数据先进行排序，然后再过滤。

```
[root@xinya ~]# cut -d : -f 7 /etc/passwd | sort | uniq
/bin/bash
/bin/sync
/sbin/halt
/sbin/nologin
/sbin/shutdown
```

下面对 uniq 命令的用法加以说明。

【语法】**uniq ［ 选项 ］**

选项说明如下：

-i：忽略大小写的不同。

-c：进行计数，统计被过滤内容的条数。

【示例】统计截至目前有多少用户登录过系统以及每个用户的登录次数。

```
[root@xinya ~]# last | cut -d ' ' -f 1 | sort | uniq -c
      6 admin1
      1 admin2
      1 ciyingch
      1 guowenmi
     33 reboot
    137 root
      8 user1
      1 wtmp
      4 zhangkai
      1 zhangshu
```

7.4.6　数量统计命令 wc

wc 命令主要用于对文件进行行数和字符数的统计。

【示例】统计/etc/passwd 文件中包含的行数、单字数和字符数。

```
#wc /etc/passwd
[root@xinya ~]# wc /etc/passwd
  43     60    1852     /etc/passwd
  ↑      ↑      ↑             ↑
 行数  单字数  字符数        文件名
```

wc 命令默认会统计文件所包含的行数、单字数和字符数。

下面对 wc 命令的用法加以说明。

【语法】**wc ［ 选项 ］ 文件名**

选项说明如下：

-l：统计文件所包含的行数。

-w：统计文件所包含的单字数。

-m：统计文件所包含的字符数。

7.4.7　输出备份命令 tee

输出备份命令 tee 的作用是将由标准输入读入的数据输出至标准输出和特定的文件，即将标准输出的内容同时保存至文件。

【语法】**命令 A | tee 文件名|命令 B**

命令 A 的标准输出通过管道传递给 tee 命令，作为 tee 命令的标准输入。tee 命令会在将其保存到所指定的文件中的同时，将其通过管道传递给命令 B。

【示例】查询所有账号的最近登录状态，并将查询结果保存至 file1 文件中，同时将通过终端登录过系统的用户显示出来。

```
[root@xinya ~]# lastlog | tee file1 | grep -v "Never logged in"
Username         Port     From             Latest
root             pts/1    192.168.1.108    Thu Feb  2 21:37:19 +0800 2012
user1            pts/3    192.168.1.108    Thu Jan 19 22:07:39 +0800 2012
admin1           :0                        Fri Jan 20 19:49:29 +0800 2012
admin2           pts/3    192.168.1.108    Fri Jan 20 19:50:52 +0800 2012
```

tee 命令将输出保存至文件的操作是一种覆盖式的操作。当需要使用追加方式操作时可以使用 tee 命令的-a 选项。

7.4.8 内容替换命令 tr

内容替换命令 tr 用于替换和删除字符串。

```
[root@xinya ~]# date
Thu May  5 10:15:04 CST 2011
[root@xinya ~]# date | tr ":" "-"
Thu May  5 10-15-09 CST 2011
```

在上例中 tr 命令将“:”号替换为“-”。

```
[root@xinya ~]# who
root     pts/1        Feb  2 21:37 (192.168.1.108)
[root@xinya ~]# who | tr '[a-z]' '[A-Z]'
ROOT     PTS/1        FEB  2 21:37 (192.168.1.108)
```

在上例中 tr 命令将所有小写字母替换为大写字母。

下面对 tr 命令的用法加以说明。

【语法】**tr [选项]**

选项说明如下：

-d 字符串：用于删除行中的字符串。

-s 字符串：用于删除重复的字符串。

```
[root@xinya ~]# ifconfig | grep "inet addr" | cut -d : -f 2 |tr -d Bcast
| tr -d Mk
192.168.1.109
127.0.0.1
```

7.4.9 文档合并命令 join

文档合并命令 join 用于将两个不同的文档合并为一个文档。

join 命令以行为单位进行操作，其工作机制是，在文件 1 和文件 2 的行中定义一个特定的字段，join 会使用文件 1 的第一行的特定字段比较文件 2 的所有行的特定字段，若这两个特定字段的值相同，则 join 会认为这两行为匹配行，并将这两行合并为一行，并且特定字段值会出现在行首。

下面以/etc/passwd 文件和/etc/shadow 文件为例来说明。

```
[root@xinya ~]# cat /etc/passwd | head -n 1
root:x:0:0:root:/root:/bin/bash
[root@xinya ~]# cat /etc/shadow | head -n 1
root:$1$IltuxymH$uIds/Z7SIQDUypwCtuzVI.:15047:0:99999:7:::
```

/etc/passwd 的文件结构为“用户名：密码：UID：GID：用户信息：用户主目录：登录 Shell”，/etc/shadow 的文件结构为“用户名：加密密码：密码相关属性”。若需要将两个文件合并，将同一用户的配置和密码信息合并为一行时，就需要利用 join 命令的功能了。

从结构上可以定义/etc/passwd 文件中定义用户名的字段为特定字段，/etc/shadow 文件中定义用户名的字段为特殊字段。join 会先用/etc/passwd 文件第一行的特殊字段“root”去比较/etc/shadow 文件中的每一行的特殊字段，若两个字段的值相同，则 join 会认为这两行相匹配，就将其合并为一行，并将特定字段值置于行首。

```
[root@xinya ~]# join -t : /etc/passwd /etc/shadow |head -n 5
root:x:0:0:root:/root:/bin/bash:$1$IltuxymH$uIds/Z7SIQDUypwCtuzVI.
:15047:0:99999:7:::
bin:x:1:1:bin:/bin:/sbin/nologin:*:15047:0:99999:7:::
daemon:x:2:2:daemon:/sbin:/sbin/nologin:*:15047:0:99999:7:::
adm:x:3:4:adm:/var/adm:/sbin/nologin:*:15047:0:99999:7:::
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin:*:15047:0:99999:7:::
```

注意，在上例中 join 命令使用-t 选项来定义字段之间的分隔符为“:”，但是并未定义/etc/passwd 和/etc/shadow 两个文件的特定比较字段，这是因为在未定义特定比较字段的情况下，join 会使用第一字段作为特定比较字段。这个命令的完整用法应该是：

```
[root@xinya ~]# join -t: -1 1  /etc/passwd  -2 1 /etc/shadow | head -n 5
                     ↑    ↑                   ↑
                定义分隔符 文件 1 的特定字段     文件 2 的特定字段

root:x:0:0:root:/root:/bin/bash:$1$IltuxymH$uIds/Z7SIQDUypwCtuzVI.
:15047:0:99999:7:::
bin:x:1:1:bin:/bin:/sbin/nologin:*:15047:0:99999:7:::
daemon:x:2:2:daemon:/sbin:/sbin/nologin:*:15047:0:99999:7:::
adm:x:3:4:adm:/var/adm:/sbin/nologin:*:15047:0:99999:7:::
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin:*:15047:0:99999:7:::
```

下面对 join 命令的用法加以说明。

【语法】**join [选项] [-1 字段号] 文件名 1 [-2 字段号] 文件名 2**

选项说明如下：

-t 分隔符：定义字段间的分隔符。

-i：忽略大小写字母的不同。

-1 字段号：定义文件 1 的特定字段，默认为第一个字段。

-2 字段号：定义文件 2 的特定字段，默认为第一个字段。

【示例】将/etc/passwd 文件中的用户信息和/etc/group 文件中的用户组信息合并为一行。

```
[root@xinya ~]# join -t ':' -1 4 /etc/passwd -2 3 /etc/group | head -n 5
0:root:x:0:root:/root:/bin/bash:root:x:root
1:bin:x:1:bin:/bin:/sbin/nologin:bin:x:root,bin,daemon
2:daemon:x:2:daemon:/sbin:/sbin/nologin:daemon:x:root,bin,daemon
4:adm:x:3:adm:/var/adm:/sbin/nologin:adm:x:root,adm,daemon
7:lp:x:4:lp:/var/spool/lpd:/sbin/nologin:lp:x:daemon,lp
```

7.4.10 文件切割命令 split

文件切割命令 split 用于对文件进行分割，以便于对大容量文件进行存储和传输。

下面对 split 命令的用法加以说明。

【语法】**split [选项] 文件名 切割后的文件名前导**

选项说明如下：

-b 文件大小：定义切割后每个文件的大小，单位是 b（代表 Byte）、k（代表 KB）或 m（代表 MB），如 300m。

-l 行数：以行数为单位进行切割。

【示例】将/boot/initrd-2.6.18-194.el5.img 文件备份为 initrd-2.6.18-194.el5.img.bak，并将备份文件切割为容量为 300KB 的多个文件。

```
[root@xinya ~]# split -b 300k initrd-2.6.18-194.el5.img.bak initrdbak
[root@xinya ~]# ll -h
total 6.4M
-rw------- 1 root root 3.2M May  5 12:31 initrd-2.6.18-194.el5.img.bak
-rw-r--r-- 1 root root 300K May  5 12:31 initrdbakaa
-rw-r--r-- 1 root root 300K May  5 12:31 initrdbakab
-rw-r--r-- 1 root root 300K May  5 12:31 initrdbakac
-rw-r--r-- 1 root root 300K May  5 12:31 initrdbakad
-rw-r--r-- 1 root root 300K May  5 12:31 initrdbakae
-rw-r--r-- 1 root root 300K May  5 12:31 initrdbakaf
-rw-r--r-- 1 root root 300K May  5 12:31 initrdbakag
-rw-r--r-- 1 root root 300K May  5 12:31 initrdbakah
-rw-r--r-- 1 root root 300K May  5 12:31 initrdbakai
-rw-r--r-- 1 root root 300K May  5 12:31 initrdbakaj
-rw-r--r-- 1 root root 199K May  5 12:31 initrdbakak
```

对于这些切割后的小文件而言，若需要将其还原为一个文件，可直接使用 cat 命令。

cat 文件名 1…文件名 *n* >> 还原后的文件名

```
[root@xinya ~]# cat initrdbak* >> initrd  ←将所有切割文件还原为 initrd 文件
```

7.4.11 参数传递命令 xargs

参数传递命令 xargs 的作用是由 stdin 读入数据，并以空格和换行符为标记，将 stdin 的数据分割成若干个命令的参数，传递给命令使用。语法格式如下：

命令 A | xargs 命令 B

将命令 A 的输出结果以空格或换行符为标志切割成若干个参数作为命令 B 的参数使用。

【示例】显示最近通过终端登录系统用户的用户信息。

```
[root@xinya ~]#  lastlog |  grep -v "Never logged in" | cut -d ' ' -f 1
|xargs finger
finger: Username: no such user.
Login: root                           Name: root
Directory: /root                      Shell: /bin/bash
…
Login: user1                          Name: (null)
Directory: /home/user1                Shell: /bin/bash
…
Login: admin1                         Name: (null)
Directory: /home/admin1               Shell: /bin/bash
…
Login: admin2                         Name: (null)
Directory: /home/admin2               Shell: /bin/bash
…
```

下面对 xargs 命令的语法加以说明。

【语法】**xargs [选项] 命令**

选项说明如下：

-o：用于将 stdin 中的空格、\、"字符识别为普通字符。

-e 字符串：xargs 命令将会使用字符串以前的内容作为命令的参数。

-p：使用交互式命令执行方式。

-n *x*：*x* 为数字，定义 xargs 向命令发送几个参数。

在使用 xargs 命令时还要注意，当 xargs 后未指定命令时，默认使用 echo 命令。

以上介绍了几个常用的管道命令，在使用 bash 的管道功能时还要注意，在管道中若单独使用“-”作为命令参数时，“-”表示取前一个命令的标准输出。

7.5 bash 的其他功能

在 bash 的命令行中，除上述介绍的功能之外，还有一些小的实用功能。包括计算功能、指令替代功能、子 Shell 功能、多指令功能以及指令组功能。本节将集中介绍这些功能。

7.5.1 bash 的计算功能

bash 的计算功能是指在命令行环境下支持对整型算术表达式的计算，其实现语法有两种：

```
echo  $[ 算术表达式 ]
echo  $(( 算术表达式 ))
```

```
[root@xinya ~]# echo $[ 200+300 ]
500
[root@xinya ~]# echo $(( (3+3)/2 ))
3
```

bash 的计算功能在使用时需要注意两个方面：

（1）bash 的计算功能仅支持整数运算，不支持浮点运算。

（2）bash 的计算功能支持+、–、*、/四则混合运算。

7.5.2 bash 的指令替代功能

bash 的指令替代功能也是一种连接两个命令的方法，其语法结构有两种，分别为：

```
命令 A '命令 B' ←此时使用的是反单引号' '，按键位于 Tab 键上方
命令 A$(命令 B)
```

将命令 B 的执行结果作为命令 A 的参数使用。

【示例】在屏幕上输出“Today is 当前系统日期”。

```
#echo  “Today is 'date +%Y-%m-%d'”
[root@xinya ~]# echo "Today is 'date +%Y-%m-%d'"
Today is 2011-05-05
[root@xinya ~]# echo "Today is"$( date +%Y-%m-%d )
Today is2011-05-05
```

7.5.3 多指令功能

bash 的多指令功能是指在一个命令行中下达多个指令，允许 bash 一次执行。其语法结构为：

```
#命令 1;…;命令 n
```

命令间使用“;”进行分割。

```
[root@xinya ~]# echo "Today is " ; date "+%Y-%d-%m" ; cal;
Today is
2011-05-05
       May 2011
Su  Mo  Tu  We  Th  Fr  Sa
 1   2   3   4   5   6   7
 8   9  10  11  12  13  14
15  16  17  18  19  20  21
22  23  24  25  26  27  28
29  30  31
```

7.5.4 bash 的子 Shell 功能

bash 的子 Shell 功能与 bash 的环境设置有关，由于 bash 的环境设置的内容将在下一章讨论，所以此处仅简单说明子 Shell 的功能。

在 bash 中子 Shell 会继承父 Shell 的环境设置，但子 Shell 中对环境的修改不会影响到父 Shell。

使用子 Shell：

（命令）

```
[root@xinya ~]# x=abc;echo $x;(x=123;echo $x);echo $x
abc
123
abc
```

7.5.5 指令组功能

bash 的指令组功能是指指令组中所有命令执行之后会作为一个单一的结果输出。

{ 命令 1;…;命令 *n* }

本章中主要对 Shell 的功能进行了介绍，这些功能的组合应用会使得 Shell 的操作变得更加灵活和更有针对性。更熟练地掌握这部分功能，对于以后的 Shell 脚本程序设计是有很大帮助的。这里，我们再次强调熟练程度的重要性。在操作系统应用领域，对于一种操作系统的熟练程度越高，驾驭这种操作系统的能力就越强，而这种熟练程度的提高依靠的就是大量的使用和练习操作。

第 8 章　Shell 的环境配置

环境是操作系统中的一个重要的概念，它定义了用户与操作系统内核间接口的工作特性。每个 Shell 均会有一系列的环境配置，以确定界面语言、颜色方案、查找路径等一系列应用方案。

环境配置是利用 Shell 的变量功能来进行配置的，本章先从 Shell 的变量功能开始介绍，重点是变量的相关操作以及利用变量实现 Shell 的环境配置。

8.1 变量概述

8.1.1 变量的概念

变量这个概念在程序设计中是很常见的，用于表示在程序运行过程中其值可以改变的量，与常量相对应。在 Shell 中，变量也具有这种特性，是 Shell 提供的可供用户调用的 Shell 要素。

一个变量由变量名、变量的内存空间和变量值 3 部分构成。

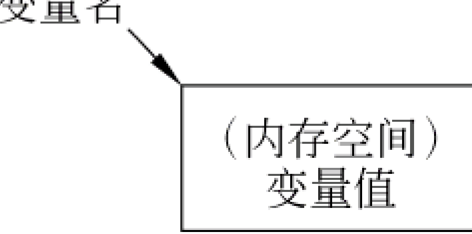


图 8.1　变量的结构

由图 8.1 可见，用户或应用程序可以通过变量名来调用存储于内存空间中的变量值，且该变量值可以根据需求的不同来进行调整，这样可大大提高对常用复杂数据的利用效率。

如我们经常会进入邮箱去读取邮件，而用户邮箱的默认存储位置为“/var/spool/mail/用户名”，当使用文件完整路径读取邮箱内容时需要使用冗长的路径信息。

```
[root@xinya ~]# cat /var/spool/mail/root
```

在 bash 中可以定义一个变量，该变量的变量值就是用户的邮箱路径，如图 8.2 所示。这样对用户邮箱的访问可直接通过调用变量即可实现，减少了用户的输入，提高了应用的灵活度。

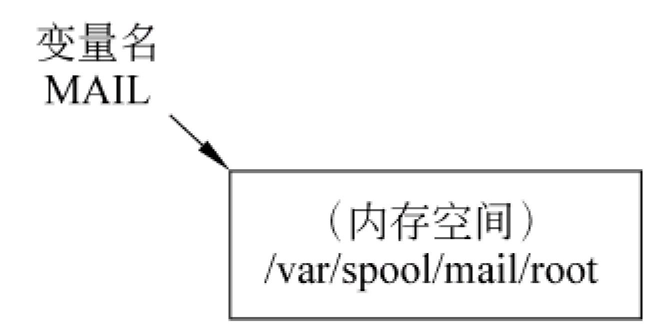


图 8.2　利用 MAIL 变量表示用户邮箱路径

```
[root@xinya ~]# cat $MAIL
```

8.1.2 变量的种类与引用

变量按其作用范围，即是否可以被子 Shell 所引用，可以分为两类：

（1）环境变量。作用于整个操作系统环境，环境变量可被当前 Shell 和子 Shell 所引用。

（2）局部变量。仅在当前 Shell 下有效，不能被其他 Shell 和子 Shell 所引用。

无论是环境变量还是局部变量，在 Shell 中用统一的引用方式。所谓引用，是指通过变量名来调用变量值的过程。引用变量的语法是“$变量名”，即在变量名前添加$符号。如刚刚使用过的“cat $MAIL”命令，就是在 MAIL 变量的前面添加$符号，表示该参数为变量而非普通参数。

8.1.3 查看变量

查看变量的方法有多种，下面分别介绍如何查看环境变量、全部变量和特定的变量。

1. 查看环境变量

查看系统中的环境变量可以使用 env 命令。（注意给出说明的变量，以后会经常使用到。）

```
[root@xinya ~]# env
HOSTNAME=xinya                          ←当前主机名变量
TERM=xterm                              ←当前终端类型变量
SHELL=/bin/bash                         ←当前 Shell 变量
HISTSIZE=1000                           ←历史命令容量变量
KDE_NO_IPV6=1
SSH_CLIENT=192.168.1.108 1634 22
SSH_TTY=/dev/pts/2
USER=root                               ←登录用户名变量
LS_COLORS=no=00:fi=00:di=00;34:ln=00;36:pi=40;33:so=00;35:bd=40;33;01:
cd=40;33;0......
KDEDIR=/usr
MAIL=/var/spool/mail/root               ←用户邮箱存储变量
PATH=/usr/kerberos/sbin:/usr/kerberos/bin:/usr/local/sbin:/usr/local/
bin:/sbin:/bin:/usr/sbin:/usr/bin:/root/bin
↑可执行文件的查找路径变量
INPUTRC=/etc/inputrc
PWD=/root
LANG=en_US.UTF-8                        ←语言环境变量
KDE_IS_PRELINKED=1
SSH_ASKPASS=/usr/libexec/openssh/gnome-ssh-askpass
SHLVL=1
HOME=/root                              ←用户主目录变量
```

```
LOGNAME=root
SSH_CONNECTION=192.168.1.108 1634 192.168.1.109 22
LESSOPEN=|/usr/bin/lesspipe.sh %s
G_BROKEN_FILENAMES=1
_=/bin/env
```

在环境变量中有几个变量要特别注意。

（1）变量 HISTORY：用于定义历史命令的容量。前面已经介绍过，默认的情况下历史命令的容量为 1000 条，而这个默认值就来源于这个变量的定义。可以通过修改该变量值来修改历史命令的容量。

（2）变量 PATH：用于定义系统中可执行文件的存储路径。当在 Shell 中输入一个命令或可执行文件名时，若未指明命令或可执行文件的存储路径，Shell 会依该变量定义的路径去查找文件或命令。

（3）变量 LANG：用于定义当前系统使用的语言环境，相关内容在后面章节将详细介绍。

2. 查看全部变量

这里的全部变量是指包括环境变量与局部变量在内的系统中的所有变量，可以使用 set 命令来查看全部变量。

```
[root@xinya ~]# set
```

set 命令的输出较多，下面介绍几个常用的局部环境变量配置文件。

（1）变量 RANDOM：是一个随机数变量，该变量以/dev/random 文件为基础，提供介于 0~32767 之间的随机数。

（2）变量 PS1：用于定义命令提示符的样式，该变量支持若干控制符号。

\d：显示“星期 月 日”。

\H：显示当前主机的完全合格域名。

\h：显示当前主机的主机名。

\t：显示时间，格式为 24 小时制的“HH:MM:SS”。

\T：显示时间，格式为 12 小时制的“HH:MM”。

\u：显示当前用户名。

\v：显示 bash 的版本信息。

\w：显示完整的工作目录。

\W：显示当前所在目录名称，使用~表示用户主目录。

\#：显示当前执行的是第几个命令。

\$：显示命令提示符，root 用户的命令提示符为#，普通用户的命令提示符为$。

对于当前的 bash 而言，其命令提示符为“[root@xinya ~]#”，含义为“[用户名@主机名 当前所在的目录命令]命令提示符”。结合上述介绍的控制符号，PS1 变量的内容应为“[\u@\h \W]\$”。当然，用户也可以根据自己的需要将命令提示符修改为其他格式（变

量的修改方法参见下一主题)。

(3)变量$:用于显示当前 Shell 的 PID。

(4) 变量?:用于显示上一个命令的返回值。这个变量很重要，在以后会经常使用该变量的功能。

3. 查看特定的变量值

查看特定的变量的变量值可使用 echo 命令直接引用该变量，以在屏幕上打印出该变量的变量值，其语法结构为:

#echo ${变量名}

或

#echo $变量名

```
[root@xinya ~]# echo $PS1
[\u@\h \W]\$
[root@xinya ~]# echo ${PATH}
/usr/kerberos/sbin:/usr/kerberos/bin:/usr/local/sbin:/usr/local/bin:/
sbin:/bin:/usr/sbin:/usr/bin:/root/bin
```

8.1.4 设置变量

所谓设置变量是指定义变量或修改变量值。变量的设置过程实际上就是被变量赋值的过程，其语法结构为:

变量名= 变量值

【示例】定义变量，并为变量赋值。

```
[root@xinya ~]# x='123456'
[root@xinya ~]# echo $x
123456
[root@xinya ~]# y='a b c d'
[root@xinya ~]# echo $y
a b c d
```

在定义变量时有几点需要注意。

(1)使用“=”为变量赋值，“=”为赋值符号，“=”号两端直接连接变量名和变量值，中间没有空格。

(2)变量名是以英文字母开头的字母和数字的组合。但变量名也支持一些特殊的符号，如$$、$?等。

(3)习惯上，环境变量名一般使用大写字母表示，局部变量名一般使用小写字母表示。对于一些涉及操作接口设置的局部变量也使用大写字母表示。

当利用“=”为变量赋值时需要注意的内容如下。

（1）变量值中若存在空格等特殊字符，需要使用转义字符" "与' '来将转义字符屏蔽。转义字符" "支持变量值中使用$，转义字符' '会将所有字符均当做普通字符处理。

（2）变量值描述中支持对变量的引用，如 ar="lang is $LANG"。在为 PATH 变量添加路径时，可以使用 PATH= "$PATH:./bin"这种方式来赋值。

（3）变量值描述中支持指令替代，如 version='uname -r'或 version=$(uname -r)

在定义变量后，被定义的变量默认是局部变量，即仅在当前 Shell 下有效。若需要将局部变量修改为环境变量时，需要借助于 export 命令。

export 命令的作用在于将一个局部变量定义为一个环境变量。

下面对 export 命令的用法加以说明。

```
export  [变量名]
export  变量名=变量值
```

export 命令单独使用时用于显示当前系统中的环境变量，其作用与 env 命令类似。

“export 变量名”命令的作用在于将一个已存在的局部变量定义为环境变量。

```
[root@xinya ~]# abc=test
[root@xinya ~]# env
HOSTNAME=xinya
…
[root@xinya ~]# export abc
[root@xinya ~]# env
abc=test
HOSTNAME=xinya
…
```

“export 变量名=变量值”命令的作用是定义一个变量，并将该变量定义为环境变量。

```
[root@xinya ~]# export use='date +%Y-%m-%d'
```

在定义了变量之后，若不再使用该变量了，可以将变量删除。删除变量的命令是 unset。

```
unset  变量名
```

```
[root@xinya ~]# unset abc
```

8.2 变量的相关操作

8.2.1 设置 Shell 的语言环境

当前 Linux 各发行版都已开始支持大多数的语系，对每个语系中的各种字符集和编码标准也实现了尽可能全面的包容。因此，在需要强调 Linux 的本地化的环境中，调整

Shell 工作时的语言环境是一个很重要的工作。

对于当前系统中都支持哪些字符集和编码标准可以使用“locale -a”命令来查看。

```
[root@xinya ~]# locale -a
aa_DJ
aa_DJ.iso88591
aa_DJ.utf8
aa_ER
aa_ER@saaho
aa_ER.utf8
aa_ER.utf8@saaho
aa_ET
aa_ET.utf8
…
```

在明确当前系统所支持的字符集的基础上，如何判断当前 Shell 环境使用的是哪种语系的字符集呢？当前 Shell 使用的字符集是通过变量来定义的，可以直接使用“locale”命令来查看与语言环境设置相关的环境变量。

```
[root@xinya ~]# locale
LANG=zh_CN.UTF-8                              ←基础语言环境定义变量
LC_CTYPE="zh_CN.UTF-8"                        ←字符识别编码使用的字符集变量
LC_NUMERIC="zh_CN.UTF-8"                      ←数字系统识别和显示使用的字符集变量
LC_TIME="zh_CN.UTF-8"                         ←时间系统识别和显示使用的字符集变量
LC_COLLATE="zh_CN.UTF-8"                      ←字符串比较与排序时使用的字符集变量
LC_MONETARY="zh_CN.UTF-8"                     ←货币信息显示所使用的字符集变量
LC_MESSAGES="zh_CN.UTF-8"                     ←系统信息显示所使用的字符集变量
LC_PAPER="zh_CN.UTF-8"                        ←页面信息显示时所使用的字符集变量
LC_NAME="zh_CN.UTF-8"                         ←姓名信息显示时所使用的字符集变量
LC_ADDRESS="zh_CN.UTF-8"                      ←地址信息显示时所使用的字符集变量
LC_TELEPHONE="zh_CN.UTF-8"                    ←电话号码信息显示时所使用的字符集变量
LC_MEASUREMENT="zh_CN.UTF-8"                  ←度量信息显示时所使用的字符集变量
LC_IDENTIFICATION="zh_CN.UTF-8"               ←身份信息显示时所使用的字符集变量
LC_ALL=                                       ←全局的语系设置环境变量
```

从示例中可见，当前系统中使用的是中文语言环境，字符集为 zh_CN.UTF-8。若需要使用英文语言环境时，可以定义各变量使用 en_US.UTF8 字符集。

在定义字符集时可以逐个定义每个变量的变量值，也可以通过定义 LANG 或 LC_ALL 变量来使得其他的语言环境变量来继承其值。即若仅指定了 LANG 或 LC_ALL 变量的值而未指定其他变量的值，则其他变量将继承 LANG 或 LC_ALL 变量值的设置。

```
[root@xinya ~]# LANG=en_US.UTF8
[root@xinya ~]# locale
LANG=en_US.UTF8
```

```
LC_CTYPE="en_US.UTF8"
…
```

注意，在当前 Shell 中对变量的定义域修改在退出 Shell 后均会归于无效。即这种修改仅在当前 Shell 中有效，退出后不保存。若需要将修改固定下来，则需要修改配置文件。

语系变量的配置文件是/etc/sysconfig/i18n。

```
[root@xinya ~]# cat /etc/sysconfig/i18n
LANG="zh_CN.UTF-8"
```

该文件用 LANG 命令定义了系统默认的基础语言环境变量为 zh_CN.UTF-8，当需要改变默认的语言环境时，可以通过修改该值来实现。

8.2.2　变量值的键盘读取

变量的赋值除可以使用“变量名=变量值”这种直接复制的方法外，还可以利用 read 命令实现将键盘输入的数据作为变量值这种赋值操作。

read 命令的基本语法结构如下：

```
read  变量名
```

【示例】将用户由键盘输入的内容复制给 test 变量。

```
[root@xinya ~]# read test
Hello welcome to Linux!          ←用户输入的内容,以 Enter 键作为输入结束标志。
[root@xinya ~]# echo $test
Hello welcome to Linux!
```

下面对 read 命令的用法加以说明。

【语法】**read　[选项]　变量名**

选项说明如下：

-p　"字符串"：打印字符串，可以将该选项作为提示信息的一种输出方法。

-t　秒数：用户输入延迟时间，若超过该时间用户未输入，则 read 退出，返回状态为非 0。

【示例】将用户由键盘输入的内容复制给变量 name，在输入前提示用户“Enter your name，Please：”。

```
[root@xinya ~]# read -p "Enter your name,Please:"  name
Enter your name,Please:xinya
[root@xinya ~]# echo $name
xinya
```

【示例】将用户由键盘输入的内容复制给变量 address，在输入前提示用户“Enter your address，Please：”，并限定用户在 30 秒内输入完成。

```
[root@xinya ~]# read -p "Enter your address,Please:"  -t 30 address
```

```
Enter your address,Please:Beijing
```

8.2.3 定义变量的类型

变量的类型包括变量的数据类型和应用类型。默认的情况下变量的数据类型是字符型，应用类型是一种可读写的本地变量。

如何修改变量的类型呢？可以使用 declare 命令。

下面对 declare 命令的用法加以说明

【语法】**declare [选项] 变量名**

选项说明如下：

declare 命令在单独使用时，其作用是显示当前系统中的所有变量，包括环境变量和本地变量。

-a：将变量的数据类型定义为数组类型。

-i：将变量的数据类型定义为整型。

-x：将变量定义为环境变量。

-r：将变量定义为只读变量。

-p：显示变量类型。

【示例】将变量定义为整型变量。

```
[root@xinya ~]# a=5;b=10;c=$a+$b;echo $c
5+10
[root@xinya ~]# declare -i c;c=$a+$b;echo $c
                    ←将变量 c 的数据类型定义为整型后，变量 c 支持数值运算。
```

【示例】将变量 c 定义为环境变量。

```
[root@xinya ~]# env
…
PWD=/root
LANG=en_US.UTF8
…
[root@xinya ~]# declare -x c
[root@xinya ~]# env
…
PWD=/root
c=15
LANG=en_US.UTF8
…
```

【示例】查看变量的类型。

declare -p 变量名

```
[root@xinya ~]# declare -p c                 ←查看变量 c 的类型
declare -ix c="15"
```

8.3 bash Shell 的操作环境

8.3.1 在 bash 下命令的查找顺序

当用户在 bash 的命令行中输入一条命令时，Shell 究竟是如何查找到该命令的呢？对于这个问题，需要分两种情况来讨论。

（1）当在命令行直接输入命令时，bash 查找命令的方法如下。

① 查找别名命令列表，以判断输入的命令是否为别名命令。若是别名命令则执行该命令，否则继续步骤②。

② 判断输入的命令是否为 Shell 的内置命令，若是则执行该命令，否则继续步骤③。

③ 按 PATH 变量定义的路径查找命令，并执行找到的第一个命令。

④ 若 PATH 变量定义的路径中不存在用户输入的命令，则提示用户“-bash: 命令: command not found”，即命令不存在，状态返回值为非 0。

（2）当在命令行以绝对路径方式输入命令时，bash 会直接去指定的目录下查询命令，若命令存在则直接执行，若命令不存在则提示用户“-bash: 命令：No such file or directory”。

8.3.2 bash 的登录与欢迎信息

bash 的登录信息是指在 bash 环境下，登录系统前用户看到的相关说明信息。这个说明信息在前面的章节中已经做过相应的说明了。

```
CentOS release 5.5 (Final)
Kernel 2.6.18-194.el5  on an i686
```

这个登录信息实际上是记录在/etc/issue 这个文件中，每次登录前系统会将该文件中的内容显示在终端界面上。

```
[root@xinya ~]# cat /etc/issue
CentOS release 5.5 (Final)
Kernel \r on an \m
```

/etc/issue 文件允许用户自定义，同时该文件中提供了一系列的控制参数，文件默认使用了\r 与\m 两个参数。该文件中所使用的参数及其含义如表 8.1 所示。

表 8.1　/etc/issue 文件中的参数及其含义

参　数	含　义
\d	显示本地端的时间和日期信息
\m	显示硬件等级信息
\o	显示主机所在的域名
\t	显示本地端的时间信息

续表

参　　数	含　　义
\v	显示 os 版本信息
\l	显示当前为第几个终端接口
\n	显示主机的网络名
\r	显示操作系统的版本信息，相当于 uname -r 命令的作用
\s	显示操作系统名称

除/etc/issue 这个文件外，系统还提供了一个/etc/issue.net 文件，该文件是向利用 Telnet 登录的用户提供登录信息的文件。不过，由于 Telnet 程序由于安全问题已不常用了，所以这个/etc/issue.net 文件的使用率也不是很高了。

/etc/issue 文件和/etc/issue.net 这两个文件是在用户登录系统前显示信息用的，所以称为登录信息文件。若需要在用户成功登录系统后向用户提供附加说明信息，可以利用/etc/motd 文件。

/etc/motd 文件中记录的内容将在用户成功登录系统后被显示出来，可以利用这个文件来提示用户相关的注意事项。

8.3.3　bash 的环境变量配置文件

在 bash 中，默认存在了一系列已配置好的环境变量。这些环境变量是如何定义的？如何将用户自定义的变量保存下来在每次系统启动后均可以继续使用呢？这就需要用到环境变量配置文件了。

Linux 系统中的环境变量配置文件分为两类：

（1）全局环境变量配置文件/etc/profile。

（2）用户环境变量配置文件~/.bash_profile、~/.bash_login 和~/.profile。

全局环境变量配置文件/etc/profile 用于设置供这个系统（包括所有用户）可使用的环境变量。该配置文件会被每个用户的登录 Shell 所读取。

可以通过 cat 命令直接查看该文件的内容，该文件中使用了许多结构控制语句，相关内容将在后续内容中陆续学习到。

/etc/profile 文件的主要作用是：

（1）定义 PATH 变量：根据登录用户的 UID 来定义 PATH 变量中是否应该包括/sbin 这个系统命令目录。

（2）定义 MAIL 变量：根据登录用户的账号名称将用户的邮箱定义为“/var/spool/mail/账号名称”这个值。

（3）定义 USER 变量：根据登录账号 UID 设置该值。

（4）定义 HOSTNAME 变量：根据 hostname 命令定义该变量的值。

（5）定义 HISTORY 变量：设置历史命令的容量，CentOS 5.x 默认为 1000 条。

（6）调用外部文件来完善环境设置。

/etc/profile 文件所调用的外部文件如下：

（1）/etc/inputrc：用于设置 bash 环境下的快捷键。/etc/profile 文件中使用“INPUTRC=/etc/inputrc” 命令来实现对该文件的调用。

（2）/etc/profile.d/*.sh：这是一个脚本的集合，只要用户对这些脚本有读取的权限，/etc/profile 就会调用并执行该脚本。这些脚本定义了 bash 操作接口的颜色、语系和别名命令等。

（3）/etc/sysconfig/i18n：这个文件由/etc/profile.d/lang.sh 脚本调用的，用于定义 bash 默认使用何种语系编码。

用户环境变量配置文件用于在用户登录系统时设置环境变量，该环境变量是在全局环境变量设置的基础上针对当前用户的需要进行的设置。

环境变量配置文件有~/.bash_profile、~/.bash_login 和~/.profile，登录 Shell 会按顺序依次读取。

登录 Shell 首先会查找~/.bash_profile 文件，若该文件存在，则调用该文件对环境变量的设置，若该文件不存在，则再依次查找~/.bash_login 文件和~/.profile 文件，若后两个文件也不存在，则继续下一步登录操作，用户未进行独立的环境变量设置。

用户环境变量配置文件~/.bash_profile、~/.bash_login 或~/.profile 的主要作用如下。

（1）定义 PATH 变量，并将 PATH 转换为环境变量。

（2）判断用户主目录下是否存在~/.bashrc 文件，若有，则在当前 shell 下执行该配置文件“. ~/.bashrc”。

~/.bashrc 文件的作用如下。

（1）定义 bash 中的别名命令。

（2）在一些 Linux 发行版中~/.bashrc 文件会调用/etc/bashrc 这个文件，实现以下功能。

① 依据不同的 UID 定义 umask 变量值。

② 依据不同的 UID 定义 PS1 变量值。

③ 调用/etc/profile.d/*.sh 脚本文件。

注意，/etc/bashrc 是 Red Hat Linux 所特有的文件，在 CentOS 中也存在相应的应用。

除上述介绍的配置文件外，系统中还有一些配置文件会对 bash 的工作环境产生影响，这些文件有：

/etc/man.config：用于定义 man 命令的相关查询环境。

~./bash_history：用于保存历史命令列表。

~/.bash_logout：退出 bash 时需要执行的相关操作可以保存在这个文件中。

对于用户自定义的环境变量可以保存在用户环境变量配置文件中，建议（除默认环境变量不适合应用要求外）最好不要修改全局环境变量配置文件。

8.3.4 终端属性的设置

Linux 默认提供了 6 个虚拟终端，分别适用 tty1~tty6 来表示，这 6 个虚拟终端的设备文件为/dev/tty[1-6]。另外还存在一个 tty0（/dev/tty0），表示的是当前终端。

终端默认已设置好了相关的工作属性，可以使用“stty -a”命令来查看终端属性设置。

```
[root@xinya ~]# stty -a
speed 38400 baud; rows 24; columns 80; line = 0;
intr = ^C; quit = ^\; erase = ^?; kill = ^U; eof = ^D; eol = <undef>;
eol2 = <undef>; swtch = <undef>; start = ^Q; stop = ^S; susp = ^Z; rprnt
= ^R;
werase = ^W; lnext = ^V; flush = ^O; min = 1; time = 0;
-parenb -parodd cs8 -hupcl -cstopb cread -clocal -crtscts -cdtrdsr
-ignbrk -brkint -ignpar -parmrk -inpck -istrip -inlcr -igncr icrnl ixon
-ixoff
-iuclc -ixany -imaxbel -iutf8
opost -olcuc -ocrnl onlcr -onocr -onlret -ofill -ofdel nl0 cr0 tab0 bs0
vt0 ff0
isig icanon iexten echo echoe echok -echonl -noflsh -xcase -tostop -echoprt
echoctl echoke
```

该命令显示一系列终端属性设置，这些属性很多，下面介绍几个常用的属性的含义。

intr = ^C： intr 表示向正在运行的程序发送中断信号，在终端中可以使用 Ctrl+C 来实现（^表示 Ctrl 键）。

quit = ^\： quit 表示退出当前正在运行的程序，在终端中可以使用 Ctrl+\来实现。

eof = ^D： eof（End of file），表示输入结束，在终端中可以使用 Ctrl+D 来实现。

erase = ^?： erase 表示删除字符，在终端中可以使用 Ctrl+? 来实现。

kill = ^U： kill 表示清空当前命令行，在终端中可以使用 Ctrl+U 来实现。

stop = ^S： stop 表示停止屏幕输出，在终端中可以使用 Ctrl+S 来实现。

start = ^Q： start 表示重启前一个退出的程序，可以重新启动屏幕输出，在终端中可以使用 Ctrl+Q 来实现。

susp = ^Z： susp 表示向当前正在运行的程序发送终端停止信号，在终端中可以使用 Ctrl+Z 来实现。

对于终端环境的属性也可通过 stty 命令来自行修改和定义，其语法结构为：

stty　属性　快捷键

```
[root@xinya ~]# stty stop ^p
[root@xinya ~]# stty -a
…
eol2 = <undef>; swtch = <undef>; start = ^Q; stop = ^P; susp = ^Z; rprnt
= ^R;
…
```

鉴于终端属性应具有一致性和稳定性，所以建议不要轻易修改终端属性。

在使用终端过程中还会遇到一些输入/输出功能设置，这些功能可以使用 set 命令来定义。

下面对 set 命令的用法加以说明。

【语法】**set**　[选项]

选项说明如下：

-u：设置使用未定义变量时会显示错误信息，默认终端未启用该功能。

-v：设置输出信息前会显示信息的原始内容，默认终端未启用该功能。

-x：设置命令在执行前会显示命令的内容，默认终端未启用该功能。

-c：设置在使用输出重定向（>）时，若目标文件已存在时，目标文件的内容不会被覆盖，即采用追加的方式重定向，默认终端未启用该功能。

8.4 命令的条件式执行

在 bash 执行命令时，除直接执行外，bash 还提供了一种条件判断功能，利用该功能可依一定的条件控制命令是否执行。

bash 提供的判断条件有两个：&&（与条件）与||（非条件）。

8.4.1 &&（与条件）控制

&&（与条件）控制结构的语法为：

命令 A && 命令 B

其含义为若命令 A 成功执行（返回值为 0，echo $? 值为 0），则执行命令 B；若命令 A 未成功执行（返回值为非 0，echo $? 值不为 0），则不执行命令 B。

【示例】判断~/lab 目录中是否有 file1 文件，若有则将 file1 文件重命名为 test1。

```
[root@xinya /]# ls ~/lab/file1 && mv ~/lab/file1 ~/lab/test1
ls: /root/lab/file1: 没有那个文件或目录              ←ls 命令有错误输出
[root@xinya /]# ls ~/lab/file1 &> /dev/null && mv ~/lab/file1 ~/lab/test1
[root@xinya /]# touch ~/lab/file1 ;ll ~/lab/file1
-rw-r--r-- 1 root root 0 05-10 20:48 /root/lab/file1    ←有 file1 文件
[root@xinya /]# ls ~/lab/file1 &> /dev/null && mv ~/lab/file1 ~/lab/test1
[root@xinya /]# ll ~/lab
…
-rwxr-xr-x 1 root root  76 2012-01-19 sh11.sh
-rw-r--r-- 1 root root   0 05-10 20:48 test1     ←file1 文件被改名
```

8.4.2 ||（非条件）控制

||（非条件）控制结构的语法为：

命令 A || 命令 B

其含义为若命令 A 成功执行（返回值为 0，echo $? 值为 0），则不执行命令 B；若命令 A 未成功执行（返回值为非 0，echo $? 值不为 0），则执行命令 B。

【示例】判断~/lab 目录中是否有 file2 文件，若没有则创建 file2 文件。

```
[root@xinya /]# ls ~/lab/
sh01.sh  sh03.sh  sh05.sh  sh07.sh  sh09.sh  sh11.sh
sh02.sh  sh04.sh  sh06.sh  sh08.sh  sh10.sh  test1
[root@xinya /]# ls ~/lab/file2 || touch ~/lab/file2
ls: /root/lab/file2: 没有那个文件或目录
[root@xinya /]# ls ~/lab/
file2    sh02.sh  sh04.sh  sh06.sh  sh08.sh  sh10.sh  test1
sh01.sh  sh03.sh  sh05.sh  sh07.sh  sh09.sh  sh11.sh
```

8.4.3 &&与 || 的联合使用

当需要将&&与||联合使用时，需要注意&&与||的应用顺序。

命令 A && 命令 B || 命令 C

【示例】判断系统中是否存在 shenghao 这个用户，若存在则锁定该用户登录，若没有则创建该用户。

```
[root@xinya /]#  cut -d : -f 1 /etc/passwd |grep shenghao  && passwd -l
shenghao || useradd shenghao
shenghao
Locking password for user shenghao.
passwd: Success
```

本节针对 Shell 的环境配置展开了讨论。Shell 环境的定义是以变量的应用为基础的，灵活地使用变量与配置文件是定义个性化 Shell 环境的重要手段。

第 9 章　Linux 文件系统管理

文件系统是操作系统中非常重要的管理结构，它涉及操作系统如何利用磁盘、数据的读写方式等一系列数据存储特性，是现代操作系统中数据存储的基础。

Linux 操作系统支持多种文件系统结构，表 9.1 中列出了 Linux 中常见的文件系统。

表 9.1　常见文件系统类型

文件系统	文件系统描述
ext2	第二扩展文件系统，是早期大多数 Linux 发行版默认使用的文件系统，当前 Linux 操作系统主要使用的是 ext2 文件系统的升级版本——ext3 文件系统
ext3	第三扩展文件系统，是 ext2 文件系统的升级版本。ext3 文件系统是在 ext2 文件系统的基础上增加了日志功能，以提高文件系统的故障恢复能力。ext3 文件系统是当前大多数 Linux 发行版默认使用的文件系统。CentOS 5.x 默认使用的就是 ext3 文件系统
Reiserfs	Reiserfs 文件系统是在 Linux 扩展文件系统的基础上演化而来的一种具有日志功能的文件系统，其特点在于高效处理各种容量的文件
Jfs	是由 IBM 开发的一个日志文件系统，主要用于 IBM 服务器
XFS	由 Silicon Graphics 为其 IRIX 操作系统开发的高性能日志文件系统，后来被移植到 Linux 内核上。XFS 特别擅长处理大文件，同时提供平滑的数据传输

当前 Linux 正在转向日志式文件系统，如 ext3、Reiserfs 及 XFS。日志文件系统因其对文件系统操作过程有详细的记录，所以在文件系统故障恢复方面有良好的优势而被广泛采用。

本章讨论的文件系统是 ext 文件系统，当前主流的 ext 文件系统是 ext3。由于 ext3 是在 ext2 文件系统的基础上通过添加日志功能演化而来，所以本章从 ext2 文件系统的特性开始介绍，以便更深入的了解该文件系统。

9.1　认识 ext2 文件系统

文件系统是操作系统存储的基础，是存储设备与操作系统之间的界面。操作系统利用文件系统来组织和使用硬件存储设备。文件系统与操作系统和存储设备的关系示意如图 9.1 所示。

文件系统是操作系统通过格式化操作在硬件存储设备上添加的一层逻辑控制结构，而一个硬件存储设备（硬盘）可以通过分区的方式来容纳不同的文件系统。一个分区可以是一个文件系统，而操作系统的格式化工作也是针对分区来操作的。

注意，不同的操作系统所使用的文件系统是不同的。如 Windows 默认使用的是 NTFS 文件系统，早期的 Linux 默认使用的是 ext2 文件系统。一个操作系统系统能否识别一种文件系统，进而利用该文件系统读写存储设备取决于这个操作系统对文件系统支持的广泛程度。Linux 对文件系统有很好的支持，支持大多数常见的文件系统。

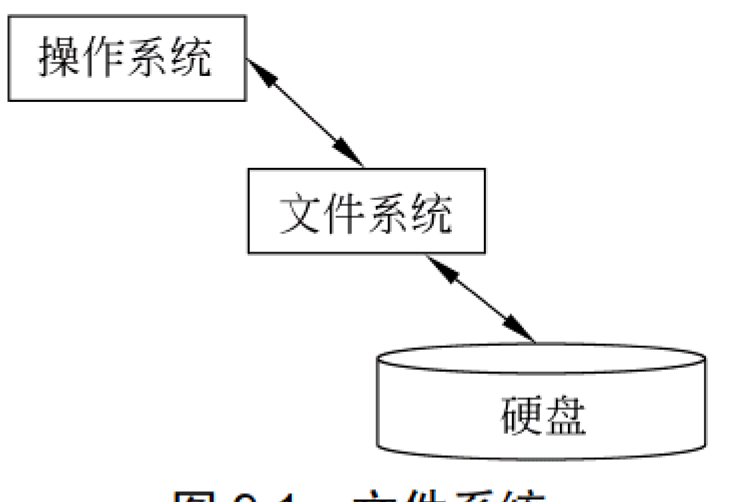


图 9.1　文件系统

在系统管理中，由于文件系统是以分区为单位的，所以有时也将一个分区称为一个文件系统。

文件系统本身是用于储存数据的，数据在文件系统中是以文件的形式表现出来的。文件从本质上可以分为两个部分：

（1）文件的属性与权限。

（2）文件的数据内容。

ext2 文件系统采用“块”这种方式存储数据，数据均存储于“块”中。对于文件而言，ext2 文件系统采用将文件的两个部分分开存储的策略，如图 9.2 所示。

（1）文件的属性与权限内容通常存储于索引节点（inode）中。

（2）文件的数据内容通常存储于称为数据区块（data block）的“块”中。

ext2 文件系统中还会有一个超级块（super block），用于记录 inode 与 data block 的数量信息。

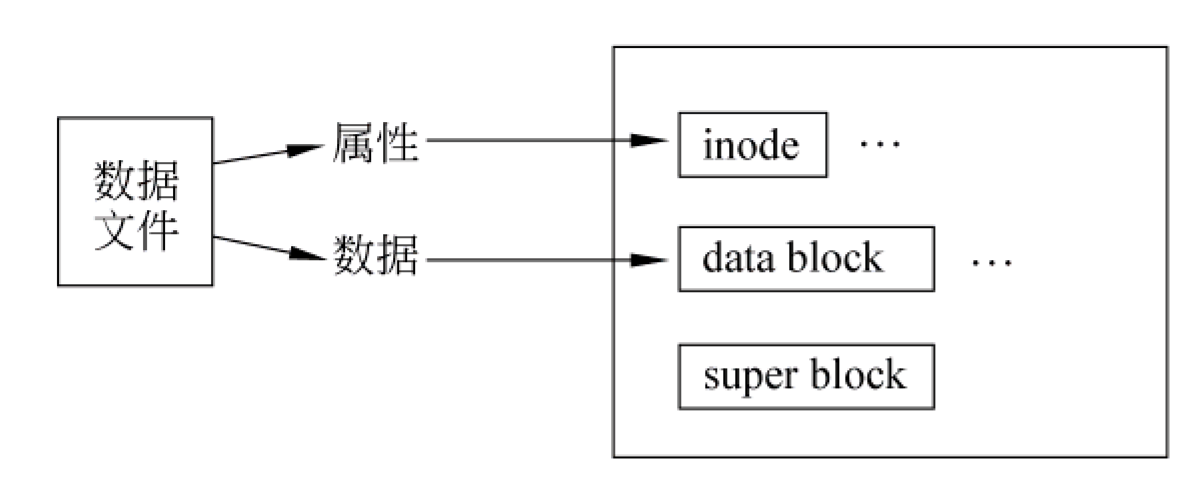


图 9.2　文件的分开存储

super block：用于记录文件系统的整体信息，包括 inode 与 data block 的总量、使用量、剩余量和文件系统格式等信息。

inode：用于记录文件的属性、文件占用的 data block 的编号。每个 inode 都会有一个编号，每个文件都必须占用一个 inode。

data block ：记录文件的数据内容。一个文件依据其大小可能会占用 1 个到多个 data block。每个 data block 都有一个编号。

ext2 采用的是一种索引式结构来查找文件。在文件系统中，每个文件均占用一个 inode，inode 中记录着文件的属性信息、文件的数据所占用的 data block 的编号。

当需要查找文件的数据内容时，只要找到文件的 inode 中的记录，就可以根据该记录中的 data block 编号定位并读取文件所占用的 data block 中的内容了。

9.1.1　ext2 文件系统中的块组

ext2 文件系统在创建时（格式化时）就已经定义好了 inode 和 datablock 的数量。这

些 inode 和 data block 统一由 super block 管理。

当分区容量很大时，直接使用 super block 管理 inode 和 data block 会降低文件系统的查找效率。为解决这个问题，ext2 文件系统引入了块组（block group）的结构。

块组（data block）是文件系统中的一种逻辑结构，每个文件系统均由若干个块组组成，每个块组都包含独立的 super block、inode 和 data block 结构。引入块组结构后，文件系统结构就演变为如图 9.3 所示的结构。

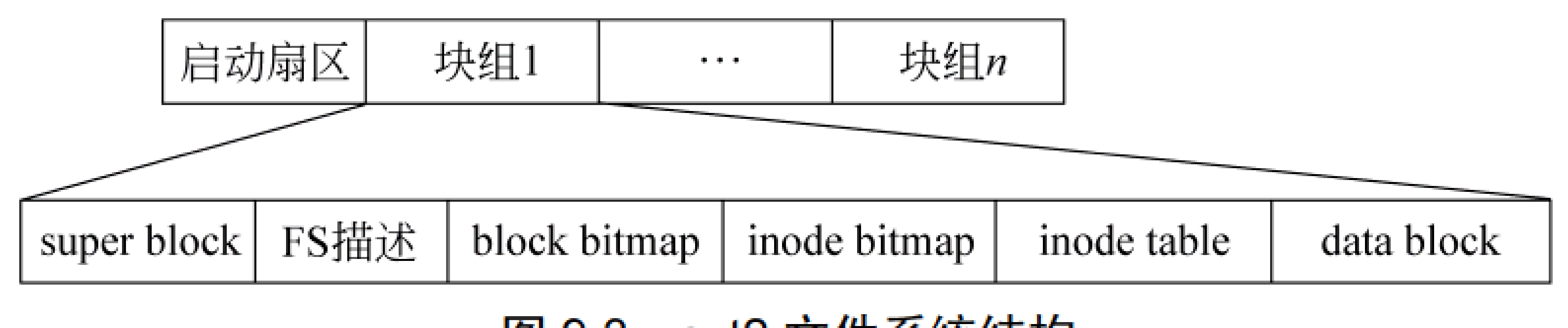


图 9.3　ext2 文件系统结构

（1）super block（超级块）用于记录整个文件系统的相关信息，包括以下具体信息。

① block 和 inode 的数量信息。

② 未使用与已使用的 block 和 inode 的数量信息。

③ block 的大小，ext2 文件系统的 block 容量可以是 1KB、2KB 或 4KB。

④ inode 的大小，inode 的容量默认为 128B。

⑤ 时间信息，包括文件系统的挂接时间、最近一次写入数据的时间、最近一次磁盘校验的时间等。

⑥ validbit 值，该值为 1 时表示文件系统未挂载，该值为 0 时表示文件系统已挂载。

super block 的容量为 1024B，理论上在每个块组中均有一个 super block，但事实上除了第一个 block group 内含有 super block 外，其他块组中很少有 super block，如果有也是第一个 block group 中的 super block 的备份。

（2）FS 描述（File System Description，文件系统描述）包含了如下信息，每个 block group 的开始与结束位置，super block、block bitmap、inode bitmap 与 data block 等字段的开始和结束位置。

（3）block bitmap（块对照表）记录了未被占用的块的编号，并回收被释放的块编号。

（4）inode bitmap（inode 对照表）记录了未被占用的 inode 编号，并回收被释放的 inode 编号。

（5）data block（数据块）：用于记录文件的内容。一个文件系统中包含了若干个 data block。

9.1.2　inode table（inode 表）

inode 是用于存储文件的属性信息和文件所占用的 data block 的编号，在 ext2 文件系统中 inode 的大小为 128B。

inode 中记录的内容如下：

（1）文件的访问模式（read/write/excute）。

（2）文件的属主和属组（owner/group）。

（3）文件的大小。

（4）文件的创建或状态改变时间（ctime）。

（5）文件的访问时间（atime）。

（6）文件的修改时间（mtime）。

（7）文件的特殊权限标志，如 SUID、SGID 和 SBIT。

（8）文件数据所占用的 data block 编号。

在 ext2 文件系统中，inode 的数量在格式化文件系统时就已经定义好了，每个 inode 的容量为 128B。文件系统中的每个文件均需要占用一个 inode，因此文件系统所能容纳文件的数量是受 inode 的数量限制的。

系统读取文件时首先找到文件的 inode，并分析 inode 中的信息以确定访问权限。当用户有权访问文件时，会通过读取文件数据所占用的 data block 编号来查找数据区块。

inode 会记录文件数据内容占用的 data block 编号，由于 inode 只有 128B，当文件占用的 data block 数量非常大时，会因为 inode 的容量限制而无法完整地记录。为此，inode 采用的是一种层级式的记录方式。

inode 的 data block 记录区域被划分为 12 个直接区块、一个间接区块、一个二层间接区块和一个三层间接区块，如图 9.4 所示。

（1）直接区块：共 12 个，每个直接区块记录了一个文件占用的 data block 编号。

（2）间接区块：记录了一个 data block 编号，该 data block 并不是文件数据占用的，而是一个专门用于记录文件数据占用了哪些 data block 编号的 data block。

（3）二层间接区块：记录了一个 data block 编号，该 data block 并不是文件数据占用的，而是一个中间记录，这个中间记录记录了还有哪些 block 记录了文件数据占用了哪些 data block 编号的 data block。

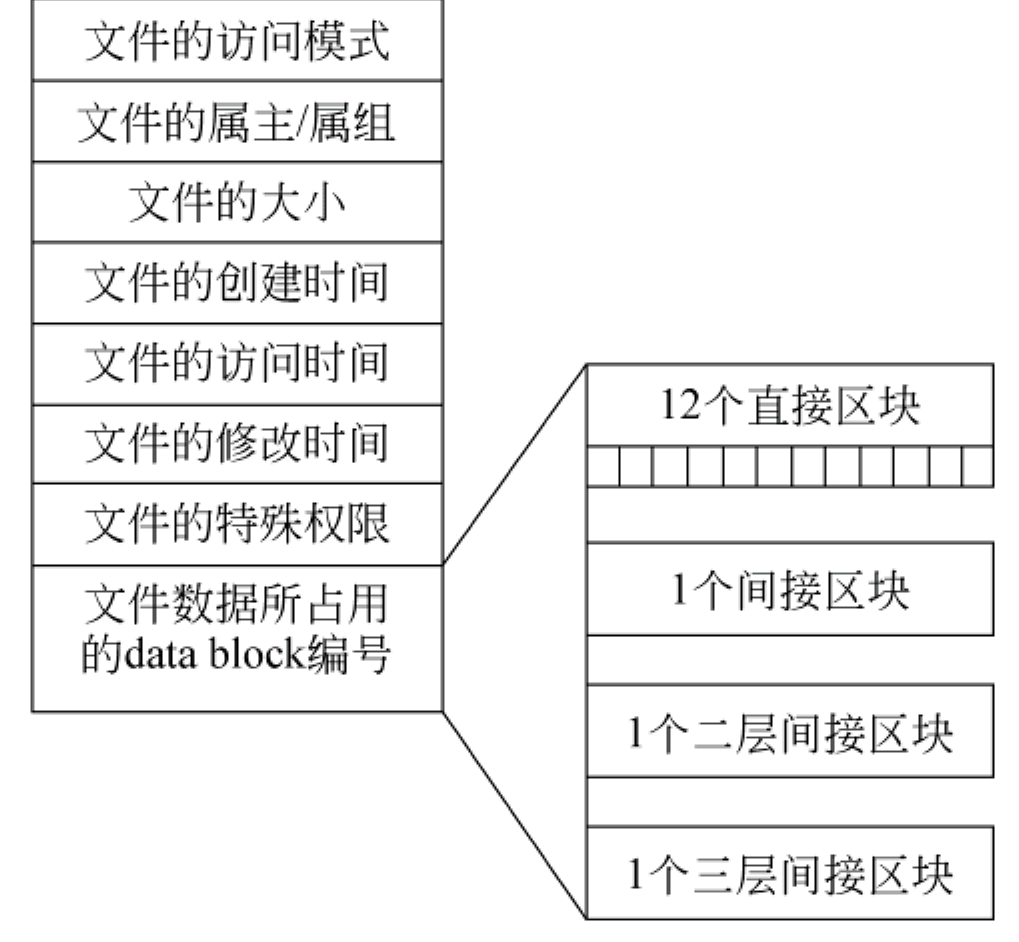


图 9.4　inode 中的 data block 记录结构

（4）三层间接区块：记录了一个 data block 编号，该 data block 并不是文件数据占用的，而是一个中间记录，这个中间记录记录了另外的中间记录，那些中间记录专门用于记录文件数据占用了哪些 data block 编号的 data block。

inode 的分层结构如图 9.5 所示。

每个编号的大小为 4B，一个 block 若为 1KB，则 inode 可包含的 data block 编号的数量为：

（1）直接区块 12 个编号。

（2）间接区块 1KB/4B=256 个编号。

（3）双层间接区块 256*256=65 536 个编号。

（4）三层间接区块 256*256*256=16 777 216 个编号。

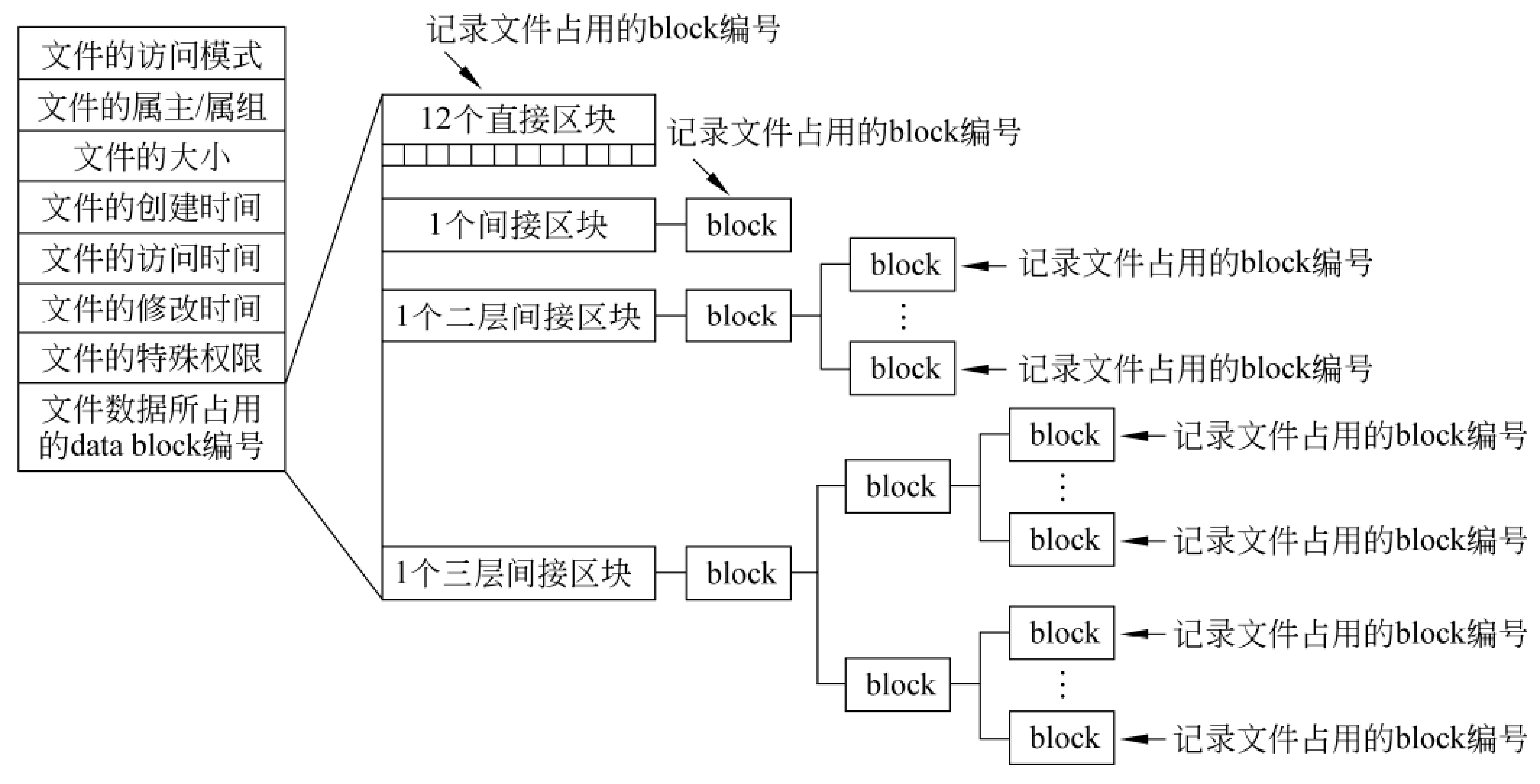


图 9.5 inode 的分层式结构

以上共计 16 843 029 个编号，若每个 data block 的容量为 1KB，则单个文件的最大容量为 16.0627GB。

9.1.3 data block（数据块）

data block 是用于存储数据的块，在 ext2 文件系统中 data block 的容量有 1KB、2KB 和 4KB 三种。data block 的容量不同会导致文件系统所支持的单一文件容量和磁盘分区容量的不同。

（1）对于 data block 为 1KB，单一文件的最大容量为 16GB，磁盘分区的最大容量为 2TB。

（2）对于 data block 为 2KB，单一文件的最大容量为 256GB，磁盘分区的最大容量为 8TB。

（3）对于 data block 为 4KB，单一文件的最大容量为 2TB，磁盘分区的最大容量为 16TB。

data block 的容量是在格式化文件系统时定义的，定义后原则上不能修改。一个 data block 中只能保存一个文件的数据，若文件过大则可能会占用多个 data block。若文件过小，则可能只占用一部分 data block，这时，剩余的 data block 空间是不能用于存储其他文件的内容的，而只能空闲不用。

9.1.4 查看文件系统信息

查看文件系统信息可以使用 dumpe2fs 命令，该命令用于显示特定文件系统信息，其语法结构为：

```
dumpe2fs 文件系统名（分区设备名）
```

当前系统已经挂接的文件系统可以使用 df 命令来查看。

```
[root@xinya /]# df
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/mapper/VolGroup00-LogVol00
                      11554008   3327584    7630048   31% /
/dev/hda1               101086     12173     83694    13% /boot
tmpfs                   127704         0     127704     0% /dev/shm
/dev/hdb1              4128020     12896    3905432    1% /home
[root@xinya /]# dumpe2fs  /dev/hda1
dumpe2fs 1.39 (29-May-2006)
Filesystem volume name:    /boot              ←文件系统名称
Last mounted on:           <not available>
…
Filesystem state:          clean              ←文件系统状态
…
Inode count:               26104              ←inode 的总量
Block count:               104388             ←data block 的总量
Reserved block count:      5219               ←保留的 block 的数量
Free blocks:               88913              ←空闲块的数量
Free inodes:               26069              ←空闲的 inode 数量
First block:               1
Block size:                1024               ←块的大小
Fragment size:             1024
Reserved GDT blocks:       256
Blocks per group:          8192               ←每个块组中包含的块的数量
Fragments per group:       8192
Inodes per group:          2008               ←每个块组中包含的 inode 的数量
Inode blocks per group:    251
…
Group 0:  (Blocks 1-8192)                     ←块组信息
  Primary superblock at 1, Group descriptors at 2-2
  Reserved GDT blocks at 3-258
  Block bitmap at 259 (+258), Inode bitmap at 260 (+259)
  Inode table at 261-511 (+260)
  990 free blocks, 1990 free inodes, 2 directories
  Free blocks: 4643-5632
  Free inodes: 19-2008
Group 1: (Blocks 8193-16384)
  Backup superblock at 8193, Group descriptors at 8194-8194
  Reserved GDT blocks at 8195-8450
  Block bitmap at 8451 (+258), Inode bitmap at 8452 (+259)
  Inode table at 8453-8703 (+260)
  3977 free blocks, 2008 free inodes, 0 directories
  Free blocks: 12408-16384
  Free inodes: 2009-4016
  …
Group 12: (Blocks 98305-104387)
  Block bitmap at 98305 (+0), Inode bitmap at 98306 (+1)
```

```
Inode table at 98307-98557 (+2)
5830 free blocks, 2008 free inodes, 0 directories
Free blocks: 98558-104387
Free inodes: 24097-26104
```

下面对 dumpe2fs 命令的用法加以说明。

【语法】**dumpe2fs [选项] 文件系统名称（分区设备文件名）**

选项说明如下：

-b：列出保留为坏道的部分。

-h：列出 super block 内的数据。

9.1.5 ext2 文件系统中的目录

前面已经介绍，每个文件都有一个 inode，通过文件的 inode 可以找到文件占用的 data block 以读取其中的内容。

文件的 inode 可以使用 ls 命令的-i 选项来查看。

```
[root@xinya lab]# ls
file2  sh01.sh  sh02.sh  sh03.sh  sh04.sh  sh05.sh  sh06.sh  sh07.sh
sh08.sh  sh09.sh  sh10.sh  sh11.sh  test1
[root@xinya lab]# ls -i
1115263 file2    1115245 sh02.sh    1115246 sh04.sh   1115223 sh06.sh
1115249 sh08.sh  1115255 sh10.sh    1115254 test1     1115224 sh01.sh
1115248 sh03.sh  1115226 sh05.sh    1115250 sh07.sh   1115252  sh09.sh
1115256 sh11.sh
```

在 ext2 文件系统中，目录与普通文件一样都要占用一个 inode，同时目录也要占用 data block，目录的 inode 和 data block 中记录的信息如下。

（1）目录的 inode 中记录了目录的相关属性信息和目录所占用的 data block 编号。

（2）目录的 data block 中记录了该目录下的文件名和子目录名，以及与之对应的 inode 编号。

在 Linux 中访问文件是利用文件名标识的，而 Linux 的实际工作过程是向目标文件所在的目录查询文件的 inode 编号，根据该 inode 编号查询 data block 编号，进而获得文件的实际数据内容。

9.2 文件系统的日志功能

ext3 文件系统是在 ext2 文件系统的基础上添加日志功能而实现的，当前在各种 Linux 发行版中 ext3 文件系统被广泛采用。

那么这个日志功能对于一个文件系统的好处究竟表现在哪呢？这需要从 ext2 文件系统的文件创建过程谈起。

在 ext2 文件系统中，新建一个文件时要完成一系列的操作，具体包括如下操作。

（1）首先要读取文件所在目录的 inode 节点信息，判断用户是否有权对该目录进行写入操作，如果用户没有该权限，则提示用户“Permission denied”（权限不足）；如果用户有该权限，则进行下一步操作。

（2）查找 inode bitmap，找到没有使用的 inode 编号分配给被创建文件，并将文件的相关属性信息写入文件的 inode。

（3）查找 block bitmap，找到未使用的 block 编号，将数据写入 data block。

（4）更新文件的 inode 记录中关于文件占用的 data block 的编号记录。

（5）更新 inode bitmap 和 block bitmap 中关于 inode 和 data block 字段的信息。

（6）更新 super block 中关于 inode 和 data block 数量的信息。

（7）更新父目录的 data block，将“文件名--inode”信息添加至父目录的 data block 中。

以上是添加一个文件时 ext2 文件系统所要完成的操作，在这种顺序分步操作中，一旦在操作完成前发生错误（如意外宕机、强行关机等）就会产生数据的不一致问题。如在更新完 inode bitmap 和 block bitmap 之后宕机，则数据已经写入，但 super block 没有反映出变化，甚至父目录的 data block 也没有记录，导致数据根本无法查找。这仅仅是问题的一个方面，更重要的是 super block 由于不清楚 inode 和 data block 的使用量信息，导致 super block 与 inode bitmap、block bitmap 的数据冲突，使得文件系统在加载时由于数据不一致而不能被正常使用。

如何解决类似的问题呢？在 ext2 文件系统中采用的是启动检查机制。即每次启动 ext2 文件系统时会检查 super block 中的 valid bit（0 为已挂载，1 为未挂载），如果是正常关机，文件系统会被正常卸载，则 valid bit 必然为 1。如果不是正常关机，则文件系统就不会被正常卸载，valid bit 必然为 0。Valid bit 的值与文件系统状态见表 9.2 所示。

当 valid bit 为 0 时意味着文件系统是非正常卸载的，这就有可能存在数据不一致的情况。为此需要参考 File system state（文件系统状态）这个参数的值。当文件系统被修改时 File system state 的值为 noclear，而当文件系统被正常卸载时 File system state 的值会被调整为 clear。若 valid bit 为 0 且 File system state 为 noclear 时，则必然表示文件系统在被修改后因异常而宕机了，这就产生了文件系统异常。

表 9.2　文件系统状态表

valid bit	File system state	文件系统状态
0	no clean	异常
0	clean	正常
1	no clean	异常
1	clean	正常

当文件系统出现异常时，启动过程会强制进行文件系统的数据一致性检查。这个检查会针对 inode bitmap、block bitmap、super block 和每个目录的 data block 以发现不一致的数据，并修正数据。注意，这种检查并不是要修正文件数据本身，同时这种检查将耗费大量的时间。

ext2 文件系统使用 fsck 这个命令检查文件系统，其语法格式为：

```
fsck  文件系统名称
```

需要注意的是，不要在文件系统被挂载的情况下使用 fsck 命令检测文件系统，否则可能对文件系统造成破坏。

对于这种数据不一致的情况，日志式文件系统提出了一种更方便的解决方案。其基本思路是在文件系统中规划出一个 data block，该 block 用于记录写入和修订文件的步骤，当发生故障时根据该日志即可完成故障的定位和排除。其具体的工作方式如下。

（1）写入前将写入操作记入日志，以便于日志能反映即将开始的写入操作。

（2）开始将文件写入 data block，并更新文件系统的相关记录。

（3）写入完成后，在日志中记录写入完成。

当文件系统发生故障时，可以根据日志发现和恢复故障点，这就避免了因为单一故障而检查整个文件系统所带来的效率上的浪费。

ext3 日志文件系统的生成是在 ext2 文件系统的基础上添加日志功能实现的，其日志属性通过 dumpe2fs 命令可以查看。

```
[root@xinya ~] # dumpe2fs /dev/hda1
…
Journal inode:          8               ←日志的 inode
Journal backup:         inode blocks
Journal size:           4114k           ←日志的大小
…
```

9.3 文件系统的基本操作

文件系统对于用户来说是透明的，因此用户对文件系统所要关心的主要是容量、挂接等应用问题。本节首先介绍文件系统的容量控制和链接文件问题，有关文件系统的创建和挂接问题我们将在下一章集中讨论。

9.3.1 查看文件系统磁盘空间的使用情况

查看文件系统磁盘空间的使用情况可以使用 df 这个命令，直接使用该命令时会显示当前已挂接文件系统（磁盘分区）的容量使用情况。

```
[root@xinya ~]# df
Filesystem          1K-blocks   Used     Available  Use%    Mounted on
/dev/mapper/VolGroup00-LogVol00
                    11554008    3327632  7630000    31%      /
/dev/hda1           101086        12173    83694    13%      /boot
tmpfs               127704            0   127704    0%       /dev/shm
/dev/hdb1           4128020       12896  3 905432   1%       /home
```

下面对 df 命令的用法加以说明。

【语法】**df [选项] [磁盘分区名|目录名]**

选项说明如下：

-a：显示所有文件系统（分区）的容量信息。

-k：以 KB 为单位显示文件系统（分区）的容量信息。

-m：以 MB 为单位显示文件系统（分区）的容量信息。

-h：以易读的方式显示容量的单位信息，包括 KB、MB 和 GB。

-H：以 1MB=1000KB 代替 1MB=1024KB。

-T：显示分区的文件系统名称，如 ext3。

-i：显示分区文件系统 inode 的使用情况。

磁盘分区名：显示特定文件系统分区的容量信息。

目录名：显示目录所在分区文件系统的容量信息。

df 命令实际上是通过读取 super block 中的信息来反映文件系统的容量信息的。

9.3.2 查看文件或目录所占用磁盘空间的情况

查看文件或目录所占用的磁盘空间的容量，使用 du 命令。

du 文件名|目录名

```
[root@xinya ~]# du  /boot/grub/grub.conf
1       /boot/grub/grub.conf
[root@xinya ~]# du /boot
12      /boot/lost+found
258     /boot/grub
6543    /boot
```

du 命令默认使用 KB 为单位来显示容量信息。

下面对 du 命令的用法加以说明。

【语法】**du [选项] [目录名| 文件名]**

选项说明如下：

-a：列出所有的目录和文件所占用的磁盘空间容量。

-h：以易读的方式显示容量信息，所谓易读的方式是为容量信息添加合适的单位信息。

-s：列出目录所占用的磁盘容量信息，包括子目录容量，但不显示子目录内容。

-S：列出目录所占用的磁盘容量信息，不包括子目录容量。

-k：以 KB 为单位显示容量信息。

-m：以 MB 为单位显示容量信息。

9.3.3 链接文件

链接文件是一种特殊的文件，用于为一个文件提供多个访问点，每个访问点称为源文件的一个链接文件。

Linux 系统中的链接分为以下两种。

（1）硬链接（Hard Link）：是指源文件与链接文件在内容上具有一致性，且链接文件不依靠源文件而独立存在。即当源文件失效时，硬链接的链接文件仍然可以正常使用。

（2）符号链接（Symbolic Link）：是指源文件与链接文件在内容上具有一致性，但链接文件必须依靠源文件而存在，即当源文件失效时，符号链接文件也同样不能继续使用。

硬链接（Hard Link）也称为实际链接，其特点如下。

（1）源文件与链接文件使用相同的 inode 号。

（2）硬链接不会导致 inode 和磁盘空间的变化，但会在所属目录的 data block 中创建“inode --文件名”的记录。

硬链接的创建可以使用 ln 命令，其语法结构为：

ln 源文件 链接文件

```
[root@xinya test]# ll
total 4
-rw------- 1 root root 607 May 11 19:39 gfile1
[root@xinya test]# ln  gfile1  hfile1
[root@xinya test]# ll
total 8
-rw------- 2 root root 607 May 11 19:39 gfile1          ←文件链接数增加 1
-rw------- 2 root root 607 May 11 19:39 hfile1          ←文件链接数为 2
```

在创建硬链接后要注意文件链接数会发生变化。

在创建硬链接时应注意以上事项。

（1）硬链接不能跨越文件系统建立。

（2）硬链接不能链接目录。

符号链接（Symbolic Link）的特点如下。

（1）符号链接使用独立的 inode 号和 data block，data block 中指定了源文件的 inode，通过这种方法来查找到源文件。所以，当源文件失效时，链接文件的 data block 所指定的源文件为失效文件，这时链接文件实际上已转换成了一个空链接文件，处于失效状态。

（2）与硬链接不同的是，符号链接可以跨越文件系统进行创建，同时符号链接也可以针对目录进行操作。

符号链接的创建同样使用 ln 命令，其语法结构为：

ln -s 源文件 符号链接文件

```
[root@xinya test]# ll
total 8
-rw------- 2 root root 607 May 11 19:39 gfile1
-rw------- 2 root root 607 May 11 19:39 hfile1
[root@xinya test]# ln -s gfile1 sfile1
[root@xinya test]# ll
total 8
-rw------- 2 root root 607 May 11 19:39 gfile1
```

```
-rw------- 2 root root 607 May 11 19:39 hfile1
lrwxrwxrwx 1 root root   6 May 11 19:50 sfile1 -> gfile1
```

在创建符号链接后，文件的链接数不会发生变化，但符号链接的文件类型为“l”。

硬链接会导致文件的链接数发生变化，文件的链接数实际上是指文件系统中有多少个文件使用系统的 inode 节点号。有关于文件的链接数有两点需要注意。

（1）文件/目录的链接数是针对硬链接而言的。

（2）创建目录时，新创建目录会自动生成一个“.”目录作为当前目录的硬链接，所以此时目录的链接数为 2（当前目录与“.”目录），同时当前目录下会创建一个“..”的目录作为父目录的硬链接，这会导致父目录的链接数增加 1。

本章主要讨论了 Linux 的 ext2 与 ext3 文件系统的基本概念。尽管本章以文件系统为名，但仅仅介绍了文件系统的基本概念。Linux 文件系统是一个复杂而重要的主题，需要多读、多练才能掌握。深入掌握文件系统的概念、方法与控制，对系统管理工作是很有帮助的。更深入的文件系统讨论可以参考有关 Linux 内核的专著中的文件系统部分和有关文件系统的专著。

第10章　磁盘分区的创建与挂载

在了解了文件系统结构之后，本章主要讨论如何创建并使用一个文件系统。前已述及，文件系统是以磁盘的分区为基础的，创建分区并使用分区的过程实际上就是创建文件系统的过程。

在 Linux 操作系统中创建分区（建立文件系统）的过程可以分为 4 个步骤。

（1）创建磁盘分区。

（2）对分区进行格式化（创建文件系统）。

（3）检查文件系统。

（4）挂载文件系统。

接下来以上述 4 个步骤为顺序展开讨论，介绍磁盘分区的创建与挂载操作。

10.1　创建磁盘分区

当在主机中添加一块新硬盘之后，首先就要对硬盘进行分区操作，以规划文件系统的容量，确定文件系统的起始和结束位置。

在 Linux 操作系统中，对磁盘进行分区可以使用 fdisk 命令，该命令用于查看和设置硬盘的分区状况。

10.1.1　查看已有磁盘的分区状况

当前系统中存在几块硬盘，每块硬盘的分区状态如何等问题都可以通过 fdisk 命令来查看。

以下命令用于显示当前系统中的磁盘参数与每个磁盘的分区状态：

```
# fdisk  -l
```

```
[root@xinya ~]# fdisk  -l

Disk  /dev/hda : 12.8 GB, 12884901888 bytes          ←磁盘/dev/hda 的信息
255  heads ,  63  sectors/track , 1566 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

  Device Boot       Start          End       Blocks    Id  System  ←分区状况
/dev/hda1    *          1           13        104391    83  Linux
```

```
/dev/hda2              14        1566     12474472+  8e  Linux LVM

Disk /dev/hdb: 8589 MB, 8589934592 bytes
15 heads, 63 sectors/track, 17753 cylinders
Units = cylinders of 945 * 512 = 483840 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hdb1               1        8876     4193878+  83  Linux
/dev/hdb2            8877       17753     4194382+  83  Linux

Disk /dev/hdd: 8589 MB, 8589934592 bytes
15 heads, 63 sectors/track, 17753 cylinders
Units = cylinders of 945 * 512 = 483840 bytes

Disk /dev/hdd doesn't contain a valid partition table
```

从输出信息中可见，当前主机中共有 3 块硬盘，分别是/dev/hda、/dev/hdb 和/dev/hdd，每块硬盘的容量信息和分区信息均被详细列出。

fdisk -l 命令的输出信息如下。

（1）磁盘信息。

```
Disk /dev/hda: 12.8 GB, 12884901888 bytes
磁盘 /dev/hda：容量为 12.8GB，12884901888 B
255 heads, 63 sectors/track, 1566 cylinders
磁头数 255，每个磁道上有 63 个扇区，柱面数为 1566 个
Units = cylinders of 16065 * 512 = 8225280 bytes
磁盘容量=柱面数×每个磁道上的扇区数×磁头数×512B
```

（2）分区信息。

Device 分区名	Boot 启动分区	Start 起始柱面	End 结束柱面	Blocks 块数量	Id 分区类型编号	System 系统
/dev/hda1	*	1	13	104391	83	Linux
/dev/hda2		14	1566	12474472+	8e	Linux LVM

查看特定磁盘的信息与分区状态可以使用以下命令：

fdisk -l 磁盘设备文件名

```
[root@xinya ~]# fdisk -l /dev/hda

Disk /dev/hda: 12.8 GB, 12884901888 bytes
255 heads, 63 sectors/track, 1566 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
```

```
/dev/hda1   *           1          13      104391   83  Linux
/dev/hda2              14        1566    12474472+  8e  Linux LVM
```

10.1.2 使用 fdisk 命令对磁盘进行分区

fdisk 是一个交互式命令，提供了一个交互式环境以实现分区和定义文件系统类型的操作。其语法结构为：

fdisk 磁盘设备名

```
[root@xinya ~]# fdisk /dev/hdd
Device contains neither a valid DOS partition table, nor Sun, SGI or OSF
disklabel
Building a new DOS disklabel. Changes will remain in memory only,
until you decide to write them. After that, of course, the previous
content won't be recoverable.

The number of cylinders for this disk is set to 17753.
There is nothing wrong with that, but this is larger than 1024,
and could in certain setups cause problems with:
1) software that runs at boot time (e.g., old versions of LILO)
2) booting and partitioning software from other OSs
   (e.g., DOS FDISK, OS/2 FDISK)
Warning: invalid flag 0x0000 of partition table 4 will be corrected by
w(rite)

Command (m for help): ←此处输入命令，使用 m 命令获得 fdisk 命令的帮助信息
```

使用 fdisk 的 m 命令获得 fdisk 命令的帮助信息。

```
Command (m for help): m
Command action
   A    toggle a bootable flag
   b    edit bsd disklabel
   c    toggle the dos compatibility flag
   d    delete a partition               ←删除一个分区
   l    list known partition types       ←列出已知的文件系统类型及其编号
   m    print this menu                  ←打印帮助信息
   n    add a new partition              ←添加一个新分区
   o    create a new empty DOS partition table
   p    print the partition table        ←打印当前硬盘的分区列表
   q    quit without saving changes      ←退出不保存
   s    create a new empty Sun disklabel
   t    change a partition's system id   ←修改一个文件系统类型
   u    change display/entry units
   v    verify the partition table
   w    write table to disk and exit     ←保存退出
```

```
  x    extra functionality (experts only)
Command (m for help):
```

创建一个新的分区可以使用 fdisk 的 n 命令。

```
[root@xinya ~]# fdisk /dev/hdd
Command (m for help): p                    ←查看当前的分区状态

Disk /dev/hdd: 8589 MB, 8589934592 bytes
15 heads, 63 sectors/track, 17753 cylinders
Units = cylinders of 945 * 512 = 483840 bytes

  Device Boot      Start         End      Blocks   Id  System      ←无分区

Command (m for help): n                    ←创建一个新的分区
Command action
  e   extended  ←创建扩展分区
  p   primary partition (1-4)              ←创建主分区
#由于这是一块新硬盘，所以需要创建一个主分区。
p                                          ←输入 p，创建主分区
Partition number (1-4): 1                  ←分区编号
First cylinder (1-17753, default 1):
 ←起始柱面号，若未输入数据则采用默认值，默认值由 default 定义(1-17753, default 1)
Using default value 1
Last cylinder or +size or +sizeM or +sizeK (1-17753, default 17753): +2GB
                                           ←结束柱面号

Command (m for help): p                    ←使用 p 命令查看分区状态

  Device Boot      Start         End      Blocks   Id  System
/dev/hdd1              1        4135     1953756   83  Linux
Command (m for help):
```

在创建新分区时需要注意以下事项。

（1）分区类型，对于一块硬盘而言，可创建 4 个主分区或 3 个主分区和一个扩展分区以及若干个逻辑分区。

（2）起始柱面号，是指分区由哪个柱面开始，可以使用默认值，该默认值可以保证分区的连续性。也可以输入用户自定义的起始柱面号。

（3）结束柱面号，是指分区在哪个柱面结束，这个值用来定义分区的容量。由于结束柱面换算为容量时比较复杂，所以此处可以输入“+分区的容量”。分区容量的表示可以直接输入分区容量值，单位是 B，也可以以 KB、MB 或 GB 为单位输入，如 2GB。

在创建主分区之后可以使用 t 命令来定义分区的文件系统类型，t 命令接受文件系统类型编码，若要查看文件系统类型编码可以使用“l”命令。

```
Command (m for help): l                    ←查看文件系统类型编码
```

```
…
 2  XENIX root   39  Plan 9       82  Linux swap / So   c4  DRDOS/sec (FAT-
 3  XENIX usr    3c  PartitionMagic  83 Linux            c6  DRDOS/sec (FAT-
 4  FAT16 <32M   40  Venix 80286     84 OS/2 hidden C:c7  Syrinx
…
Command (m for help): t                          ←定义文件系统类型
Selected partition 1
Hex code (type L to list codes): 83              ←输入文件系统类型
```

将创建结果保存并退出。

```
Command (m for help): p                          ←显示分区状态

Disk /dev/hdd: 8589 MB, 8589934592 bytes
15 heads, 63 sectors/track, 17753 cylinders
Units = cylinders of 945 * 512 = 483840 bytes

   Device Boot      Start        End      Blocks   Id  System
/dev/hdd1               1       4135     1953756   83  Linux

Command (m for help): w                          ←将结果保存退出
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
```

用户可依上述步骤自行创建扩展分区和逻辑分区，以练习利用 fdisk 命令创建分区的操作。

10.1.3　利用 fdisk 命令删除分区

在 fdisk 命令的交互式环境中，可以使用 d 命令来删除分区。

```
[root@xinya ~]# fdisk /dev/hdd
…
Command (m for help): p                          ←打印当前的分区状态

 Device    Boot     Start         End      Blocks       Id   System
/dev/hdd1               1        4135      1953756      83   Linux
/dev/hdd2            4136        8270      1953787+     83   Linux
/dev/hdd3            8271       17753       4480717+     5   Extended
/dev/hdd5            8271       12405       1953756     83   Linux
/dev/hdd6           12406       16540       1953756     83   Linux
                                         ←系统 d 第 6 个分区

Command (m for help): d                          ←删除一个分区
Partition number (1-6): 6                        ←被删除的分区号
```

```
Command (m for help): p                    ←打印当前的分区状态

 Device    Boot      Start       End      Blocks   Id  System
/dev/hdd1               1       4135     1953756    83  Linux
/dev/hdd2            4136       8270     1953787+   83  Linux
/dev/hdd3            8271      17753     4480717+    5  Extended
/dev/hdd5            8271      12405     1953756    83  Linux
                                              ←第 6 个分区已被删除
```

分区被删除后同样需要使用 w 命令保存刚才的操作并退出。

若不想保存刚才的操作，则可以使用 q 命令退出 fdisk，但不保存操作。

使用 fdisk 命令对硬盘进行分区操作后，由于不能立刻更新硬盘分区表，所以在完成 fdisk 操作后需要重新启动操作系统，以便及时更新分区表。

10.2　对分区进行格式化

在完成分区操作后，需要将特定的文件系统加入新建的分区，才能使用分区存储文件。将文件系统加入磁盘分区的过程叫做磁盘格式化。也就是说在完成磁盘分区后紧接着应完成磁盘格式化的操作。

在进行分区格式化操作之前应首先使 Linux 操作系统内核接受新建的磁盘分区表，这个过程通过重新启动操作系统即可实现。同时，系统提供了 partprobe 命令，可强制内核更新分区表（注意，在 partprobe 命令无法使内核更新分区表时，需要重新启动操作系统才能更新分区表）。当内核接受新建的磁盘分区表后即可以使用 mkfs 命令进行分区格式化操作了。

mkfs 命令是一个命令集，会调用相关的具体格式化程序完成格式化操作。例如，格式化 ext2 文件系统时，mkfs 会调用/sbin/mkfs.ext2 这个工具程序来完成格式化操作；再如，格式化 FAT 文件系统时，mkfs 会调用/sbin/mkfs.vfat 这个工具程序来完成格式化操作。

下面对 mkfs 命令的用法加以说明。

【语法】**mkfs　[选项]　分区设备名**

选项说明如下：

-t：定义被格式化分区的文件系统类型，常用的有 ext2、ext3、vfat 和 swap 等。如果不使用-t 选项，mkfs 将会根据分区时定义的文件系统类型格式化文件系统。

【示例】将/dev/hdd1 分区格式化为 ext2 文件系统。

```
[root@xinya ~]# mkfs -t ext2 /dev/hdd1
mke2fs 1.39 (29-May-2006)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)                    ←块大小默认为 4KB
Fragment size=4096 (log=2)
```

```
244320 inodes, 488439 blocks            ←inode 和 data block 数量
24421 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=503316480
15 block groups
32768 blocks per group, 32768 fragments per group
16288 inodes per group
Superblock backups stored on blocks: 32768, 98304, 163840, 229376, 294912

Writing inode tables: done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 28 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
```

前已述及，当利用 mkfs 命令格式化文件系统时，该命令会调用/sbin/目录下相应的工具程序，所以也可以直接使用/sbin/mkfs.ext2 来完成格式化操作。

【示例】将/dev/hdd5 分区格式化为 ext2 文件系统。

```
[root@xinya ~]# /sbin/mkfs.ext2 /dev/hdd5
mke2fs 1.39 (29-May-2006)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
244320 inodes, 488439 blocks
24421 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=503316480
15 block groups
32768 blocks per group, 32768 fragments per group
16288 inodes per group
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912

Writing inode tables: done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 24 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
```

当利用 mkfs 或/sbin/mkfs.ext2 格式化文件系统时无法指定 data block 大小和 inode 数量值等具体参数，若需要定义这些值时，可以使用 mke2fs 命令。

mke2fs 命令也是用于格式化文件系统的，但该命令支持对文件系统参数的定义，比 mkfs 命令能更细致地规划文件系统参数。

下面对 mke2fs 命令的用法加以说明。

【语法】**mke2fs [选项] 分区设备名**

选项说明如下：

-b data block 大小：定义 data block 的大小，支持 1024、2048、4096 三种容量。

-i 容量值：定义多少个 data block 容量给予一个 inode。默认每两个 data block 分配一个 inode。如每 4096B 分配一个 inode，则表示为“-i 4096”。

-L 卷标：设置分区卷标。

-j：为 ext2 文件系统加入日志，形成 ext3 文件系统。

-c：检查磁盘错误，当使用一次-c 时进行快速读取测试，当使用两次-c 时进行读写测试。

【示例】将/dev/hdd6 格式化为 ext3 文件系统，且 data block 的大小为 4K，每两个 data block 分配一个 inode。

```
[root@xinya ~]# mke2fs -b 4096 -i 8192 -j /dev/hdd6
mke2fs 1.39 (29-May-2006)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
244320 inodes, 488439 blocks
24421 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=503316480
15 block groups
32768 blocks per group, 32768 fragments per group
16288 inodes per group
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912

Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 27 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
```

对于一个已存在的 ext2 文件系统而言，若需要将其转换为 ext3 文件系统时，可以直接利用 tune2fs 命令为现有的 ext2 文件系统添加日志而形成 ext3 文件系统。

下面对 tune2fs 命令的用法加以说明。

【语法】**tune2fs [选项] 分区设备名称**

选项说明如下：

-j：将 ext2 文件系统转换为 ext3 文件系统。

-l：读取超级区块信息。

-L：修改文件系统卷标。

【示例】 将/dev/hdd1 分区的文件系统由 ext2 修改为 ext3 文件系统，并添加卷标“system”。

```
# tune2fs  -j -L “system”  /dev/hdd1
[root@xinya ~]# tune2fs  -j -L "system"  /dev/hdd1
tune2fs 1.39 (29-May-2006)
Creating journal inode: done
This filesystem will be automatically checked every 28 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
[root@xinya ~]# dumpe2fs -h /dev/hdd1
dumpe2fs 1.39 (29-May-2006)
Filesystem volume name:   system           ←系统卷标
…
Journal inode:           8                 ←已添加日志
Default directory hash:   tea
Directory Hash Seed:     ea442449-84e3-498f-9c97-726b962e26eb
Journal backup:          inode blocks
Journal size:            32M
```

10.3　检查磁盘文件系统

在建立了分区和文件系统并使用一段时间后，需要检查磁盘是否有坏道，这时可以使用 fsck 和 badblocks 两个命令来检查磁盘。

fsck 命令在使用过程中如果发现任何错误文件，会将该错误文件存储于“lost+found”目录中。

10.3.1　检查与修正磁盘错误

检查与修正磁盘错误可以使用 fsck 命令，该命令是一个 root 权限命令。fsck 命令在扫描硬盘时，可能会造成部分文件系统的损坏，所以在执行 fsck 命令时，被检查的分区务必不能挂载到系统上，也就是应在卸载的状态。

下面对 fsck 命令的用法加以说明。

【语法】**fsck　[选项]　分区设备文件名**

选项说明如下：

-t：fsck 可以检查多种文件系统，不同文件系统的检查程序都在/sbin 中，可以使用“ls -l　/sbin/fsck*”查看。使用-t 选项可以要求 fsck 只查看特定的文件系统。

-A：按照/etc/fstab 的内容，将所有的设备都扫描一次（通常启动过程中会执行此命令）。

-r：当发现错误时要求用户确认是否需要修复。

-a：自动修复检查到的有问题的区域，不需要用户确认。

-y：与-a 类似，但有些系统不支持-a 选项，这种情况下可以使用-y 选项。

-C：可以在检查过程中使用进度条显示检查进度。

-f：强制检查。一般来说，如果 fsck 未发现任何 unclean 标志是不会主动内部检查的，-f 强制 fsck 进入该文件系统检查。

【示例】检查/dev/hdd5 文件系统。

```
[root@xinya ~]# fsck -C -t ext3 /dev/hdd5
fsck 1.39 (29-May-2006)
e2fsck 1.39 (29-May-2006)
/dev/hdd5: clean, 11/244320 files, 8397/488439 blocks
[root@xinya ~]# fsck -Cf -t ext2 /dev/hdd5          ←强制检查文件系统
fsck 1.39 (29-May-2006)
e2fsck 1.39 (29-May-2006)
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information
/dev/hdd5: 11/244320 files (9.1% non-contiguous), 8397/488439 blocks
```

10.3.2 检查磁盘坏道命令

检查磁盘坏道可以使用 badblocks 命令，由于 fsck 的功能已经很强大了，所以 badblocks 命令很少被使用。

下面对 badblocks 命令的用法加以说明。

【语法】**badblock [选项] 分区设备文件名**

选项说明如下：

-s：在屏幕上显示进度条。

-v：查看进度信息。

-w：使用写入方式来测试。建议不要使用此参数，尤其是被检查设备中包含文件时。

10.4 挂载分区文件系统

在对磁盘完成分区划分和格式化操作之后，还需要将新建立的分区文件系统挂载到当前文件系统才能使用该分区。前面已经提到，Linux 是一个单根文件系统，所谓挂载是将新建立的分区文件系统连接到已有的单根文件系统的过程。

本节将对分区文件系统的挂载进行讨论。

10.4.1 挂载与挂载点

如果用户使用过 Windows 系统，则可能不太容易理解挂载（mount）的概念。假如用户有光盘驱动器，要读取光盘时，在 Windows 系统下用户可能直接打开 D:或 G:就可以；但是，由于 Linux 是一个单根文件系统，设备是以文件的形式存在的。这时如果读

取光盘上的内容就需要将由光盘驱动器文件代表的光盘文件系统与现有的单根文件系统进行结合，即将光盘中的目录及文件系统放置在整个系统的目录树中，以便通过现有的路径访问光盘数据，这个过程即为挂载的过程。

在图 10.1 中存在两个目录树：原有的系统目录树及光盘目录树。把光盘挂载到系统目录树上的/mnt/cdrom 位置时，/mnt/cdrom 就是挂载点（Mount Point），挂载后的系统目录树如图 10.2 所示。

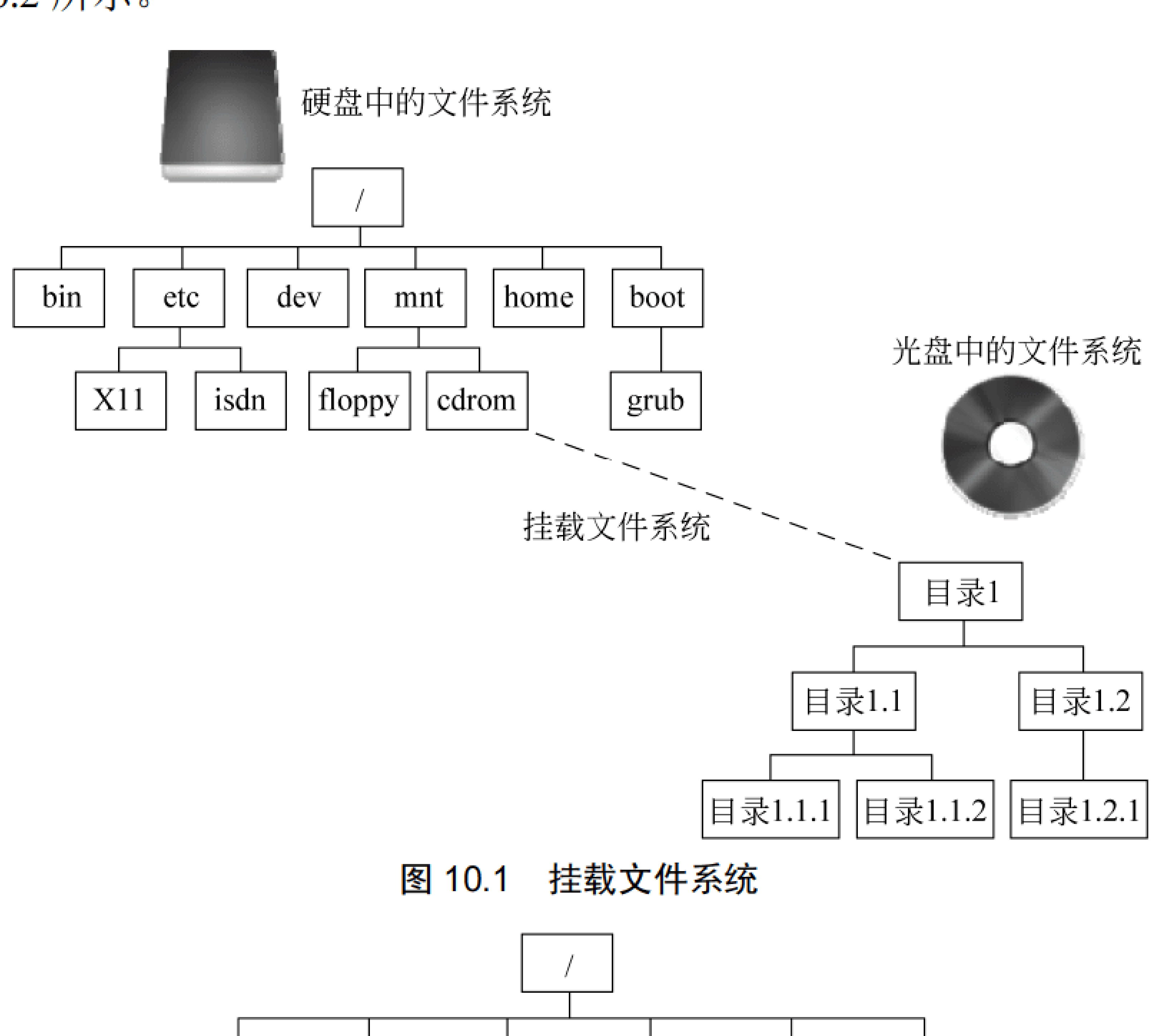


图 10.1　挂载文件系统

/
bin
etc
dev
mnt
home
boot
X11
isdn
floppy
cdrom
grub
目录1
目录1.1
目录1.2
目录1.1.1
目录1.1.2
目录1.2.1

图 10.2　挂载后的文件系统

这时用户就可以使用/mnt/cdrom 这个路径访问光盘中的文件系统了。如果用户需要使用软盘或另一块硬盘，都要经过挂载才能访问其中的数据。

磁盘的分区也需要经过挂载才能被系统使用。当开机时，系统启动的过程中就自动将硬盘的各分区挂载到设置的挂载点上了。有关启动时的自动挂载将在后面的章节中介绍。

通过上述的描述，可以给挂载下一个定义。

挂载是将不同的文件系统通过挂载点链接到一起的过程，挂载点是根文件系统下的一个已存在的目录。

10.4.2 挂载分区文件系统

挂载分区文件系统是系统管理员的工作，所以只有 root 账号有权执行挂载。要把一个设备中的文件系统目录树挂载到 Linux 文件系统目录下，可以使用 mount 命名。

下面介绍 mount 命令的基本使用方法。

【语法】**mount [-t 文件系统类型] 设备名称 挂载点**

参数说明如下：

-t 文件系统类型：指定挂载的文件系统类型。Linux 可以支持多种文件系统类型，在挂载文件系统的时候，用户要告诉系统要挂载的文件目录是什么类型。可以使用的文件系统类型有 minix、ext、ext2、ext3、msdos、hpfs、nfs、iso9660、swap 和 sysv 等。其中 ext3 是现在 Linux 主机硬盘使用的文件系统类型；msdos 是 MS-DOS 的文件系统类型；iso9660 是光盘上的文件系统类型，如果要挂载光盘驱动器，应该选 iso9660；nfs 是远程 Linux 计算机的硬盘文件系统；swap 是用在交换分区的文件系统类型。

设备名称：是指被挂载设备的设备文件名。

挂载点：是在 Linux 目录树中的一个目录，在挂载之前，挂载点目录必须要事先存在。通常光驱、软驱等设备的挂载点默认被系统定义为在/mnt 目录下的 cdrom 和 floppy 两个目录中。当然，根据 FHS，当前 Linux 将默认的挂载点位置修改到了/media 目录下。用户可以使用默认的挂载点，也可以自定义挂载点。毕竟挂载点仅仅是一个目录，一个普通目录只要事先存在就能够成为设备的挂载点。

【示例】挂载光盘。

```
[root@xinya ~]# mount /dev/cdrom /media/cdrom
mount: block device /dev/cdrom is write-protected, mounting read-only
```

注意，由于当前的 Linux 内核会自动探测被挂载设备的文件系统类型，所以常见设备的挂载可以不使用-t 选项，只在系统探测设备文件系统类型失败等错误发生时使用-t 选项。

1. 以只读方式挂载文件系统

可以在挂载文件系统时将文件系统设置为只读，在这个目录下的文件只能读取，不能写入或修改。

```
mount -r 设备名称 挂载点
```

2. 以可读写的方式挂载文件系统

挂载文件系统时设置允许读取与写入的权限，这也是 mount 命令默认的选项。

```
mount -w 设备名称 挂载点
```

3. 挂载所有设备

/etc/fsta 文件中记录了系统中应该挂载的所有设备和文件系统等信息，用户可以一次挂载/etc/fstab 文件中记录的所有设备。

```
mount  -a
```

4. 挂载记录

当前系统中所有的挂载都会记录在/etc/mtab 文件中。

```
[root@xinya ~]# cat /etc/mtab
/dev/mapper/VolGroup00-LogVol00 / ext3 rw 0 0
proc /proc proc rw 0 0
sysfs /sys sysfs rw 0 0
devpts /dev/pts devpts rw,gid=5,mode=620 0 0
/dev/hda1 /boot ext3 rw 0 0
tmpfs /dev/shm tmpfs rw 0 0
/dev/hdb1 /home ext2 rw,usrquota,grpquota 0 0
none /proc/sys/fs/binfmt_misc binfmt_misc rw 0 0
sunrpc /var/lib/nfs/rpc_pipefs rpc_pipefs rw 0 0
/dev/hdc /media/cdrom iso9660 ro 0 0
```

可以直接查看该文件，也可以使用不添加任何选项和参数的 mount 命令来查看挂载记录。

```
[root@xinya ~]# mount
/dev/mapper/VolGroup00-LogVol00 on / type ext3 (rw)
proc on /proc type proc (rw)
sysfs on /sys type sysfs (rw)
devpts on /dev/pts type devpts (rw,gid=5,mode=620)
/dev/hda1 on /boot type ext3 (rw)
tmpfs on /dev/shm type tmpfs (rw)
/dev/hdb1 on /home type ext2 (rw,usrquota,grpquota)
none on /proc/sys/fs/binfmt_misc type binfmt_misc (rw)
sunrpc on /var/lib/nfs/rpc_pipefs type rpc_pipefs (rw)
/dev/hdc on /media/cdrom type iso9660 (ro)
```

当卸载某个设备或文件系统后，对应的项就会从 mtab 文件中被删除。

5. 卸载（umount）

当挂载的文件系统使用完毕后需要卸载，系统在关机时也会自动将所有的文件系统卸载。

卸载的命令是 umount，语法为：

```
umount  设备名
```

或

umount　挂载点

可以指定设备或挂载点来卸载。卸载的文件系统必须是不在使用状态中。例如，如果正在/mnt/cdrom 目录下，要卸载光驱，umount 命令会回复设备忙的错误信息，如下所示：

```
[root@xinya cdrom]# umount /media/cdrom
umount: /media/cdrom: device is busy
umount: /media/cdrom: device is busy
```

6. 卸载特定系统类型

可以指定特定的文件系统类型进行卸载：

umount　-t　文件系统类型

7. 卸载所有文件系统

卸载所有的文件系统使用如下命令：

umount　-a

下面对 mount 命令的用法加以说明。

mount 命令用于挂载一个文件系统，该命令如果不使用任何选项或参数则会显示当前挂载的数据，而不执行挂载操作。mount 命令默认只有 root 用户才能使用。

【语法】**mount　[选项]　[设备名称]　[挂载点]**

选项说明如下：

-a：将/etc/fstab 文件内所列出的所有系统挂载。

-n：载入后不在/etc/mtab 文件中添加挂载记录。

-r：以只读的方式挂载文件系统。

-v：以简单形式显示挂载信息。

-w：以可读写的方式挂载设备，默认值。

-t 文件系统类型：指定挂载设备的文件系统类型。

下面对 umount 命令的用法加以说明。

umount 命令用于将一个文件系统卸载，是 mount 命令的反向操作。

【语法】**umount　[选项]　[设备名称|挂载点]**

选项说明如下：

-a：将/etc/mtab 文件内记录的已挂载的设备全部卸载。

-n：卸载，但不改变/etc/mtab 文件中的数据记录。

-t 文件系统类型：指定卸载设备的文件系统类型。

10.4.3 管理软驱

软盘在计算机中的使用已经有很长的历史了，尽管存在着种种缺点，但是在有些环境中还需要同软盘打交道。此处将介绍软盘的使用方法。

对于软盘的使用同样要经过挂载处理后进行。

由于软驱的设备文件名为/dev/fd0 和/dev/fd1，所以软驱的挂载命令为：

```
# mount  /dev/fd0
```

因为在/etc/fstab 中通常定义好了这两项设备的挂载信息，所以用户可以不需要指定文件系统类型及挂载点。一般来说，挂载后默认的软盘内容会在/mnt/floppy 下。而挂载软驱，第一台为/dev/fd0，若有第二台就是/dev/fd1。

如果软盘的文件系统是 FAT，Linux 另外提供一套命令以方便访问软驱的内容，这套命令就是 mtools，可以直接进入/etc/fd0 文件内容来访问软驱。mtools 命令大致就是 DOS 命令加上 m 开头。尽管软盘已经很少被使用了，但是考虑到知识结构的完整性，下面列出 mtools 命令列表，见表 10.1。有兴趣的读者可以自行参考其他资料学习。

表 10.1　常用的 mtools 命令

mtools 命令	类似的 DOS 命令	说　明
mattrib	attrib	改变文件属性
mbadblocks	chkdsk	检查磁盘错误
mcd	cd	切换目录
mcopy	copy	复制文件
mdel	del	删除文件
mdeltree	deltree	删除目录树
mdir	dir	显示目录内容
mformat	format	格式化磁盘
minfo		显示磁盘格式
mlabel	label	设置磁盘卷标
mmd	md	创建目录
mmount		挂载 DOS 磁盘
mpartition		创建磁盘分区
mrd	rd	删除目录
mmove	move	移动文件
mren	ren	更改文件名称
mtoolstest		测试 mtools 命令
mtype	type	显示文件内容
mzip		压缩文件

10.4.4 管理光驱

光盘作为大容量的可靠的存储介质一直都受到用户的普遍认可，而光盘驱动器作为计算机中读取光盘的设备也在大多数主机中被配置。

Linux 可以很好地支持光盘的使用，并且光盘也是最常用的 Linux 安装介质。

在使用光驱设备时首先要明确光驱的设备文件名。在 Linux 安装时主机中如果有光驱，安装程序将自动建立光驱的设备文件。

```
[root@xinya /]# ls -l /dev/cdrom
lrwxrwxrwx 1 root root 3 05-11 21:34 /dev/cdrom -> hdc
```

/dev/cdrom 是光驱的设备文件，该文件实际上是一个符号链接，连接至/dev/hdc。/dev/hdc 才是真正的光驱设备文件。注意，这个符号链接是系统自动建立的。

在明确了光驱设备文件名后就可以进行光驱设备挂载操作了，使用 mount 命令进行挂载，在光驱挂载期间，光驱将被锁定，即使按光驱上的弹出按钮也不能弹出光盘。

```
mount -t iso9660 /dev/cdrom 光驱挂载点
```

iso9660 是光驱设备的文件系统名称。

在不使用光驱时需要对光驱进行卸载，卸载光驱时使用 umount 命令。卸载光驱前应退出光驱文件系统。

```
umount 光驱挂载点目录|光驱设备文件名
```

卸载光驱后即可弹出光盘。弹出光驱可使用 eject 命令，使用“eject －t”命令则可以收回光驱。

10.4.5 制作 ISO 文件

ISO 是受到广泛支持的光盘镜像的标准格式，可以使用光盘刻录软件从 ISO 文件中刻录出所需的光盘。

ISO 文件不仅可以从网上下载，也可以使用 cp 命令制作 ISO 文件。命令格式如下：

```
cp /dev/cdrom ISO 文件名
```

这里 cp 命令将光驱设备文件作为源文件名，ISO 文件作为目标文件，可以将光驱中的光盘制作为 ISO 文件。注意，在使用 cp 命令制作 ISO 镜像时，光驱设备是不能挂载的。

在 Linux 中不仅可以从现有的光盘制作 ISO 文件，还可以使用 mkisofs 命令把任何系统中的文件或目录制作为 ISO 文件。命令格式如下：

```
mkisofs -r -o ISO 文件名 目录名
```

mkisofs 命令用于把系统中指定目录中的所有内容制作为 ISO 文件。

```
[root@xinya /]# mkisofs -r -o boot.iso /boot
…
INFO:   UTF-8 character encoding detected by locale settings.
Total rockridge attributes bytes: 2610
Total directory bytes: 6144
Path table size(bytes): 38
Max brk space used 18000
3422 extents written (6 MB)
```

当制作完 ISO 文件后，除可以使用刻录设备将其刻录至光盘中使用外，还可以直接将 ISO 镜像文件挂载使用。对于一些不常用的光盘而言，以镜像的形式保存在磁盘中，使用时直接挂载，是节省刻录成本的很方便的方法。

挂载镜像的命令格式如下：

mount -o loop ISO 文件名 挂载点

10.5 管理文件系统卷标

对于磁盘分区而言，除通过设备文件名来引用外，还可以通过卷标来引用。卷标是用于替代设备名称使用的一种设备标识。卷标具有意义明确、引用简单的特点。

对于 Linux 磁盘的卷标管理可以包含 3 个方面：

（1）查看设备卷标。

（2）设置设备卷标。

（3）引用卷标。

1. 查看设备卷标

使用 e2label 命令可以查看和设置卷标。

查看设备卷标的命令格式如下：

e2label 设备名

```
[root@xinya /]# e2label /dev/hdd1
system            ←/dev/hdd1 设备的卷标为 system
```

2. 设置设备卷标

为分区设置卷标的命令格式如下：

e2label 设备名 '卷标名'

```
[root@xinya /]# e2label  /dev/hdd5  'software'
[root@xinya /]# e2label /dev/hdd5
software
```

设置卷标之后，当需要知道卷标名对应的设备时，可以使用 findfs 命令。命令格式如下：

findfs　LABEL=卷标名

```
[root@xinya /]# findfs LABEL='system'
/dev/hdd1
```

3. 引用卷标

当需要使用设备且分区已经存在卷标时，可以通过卷标来引用设备。引用的方法为：

LABEL=卷标名

以挂载/dev/hdd1 为例，/dev/hdd1 的卷标为 system，则挂载方法为：

```
[root@xinya /]# mount LABEL='system'  /mnt
[root@xinya /]# mount
/dev/mapper/VolGroup00-LogVol00 on / type ext3 (rw)
proc on /proc type proc (rw)
…
sunrpc on /var/lib/nfs/rpc_pipefs type rpc_pipefs (rw)
/dev/hdc on /media/cdrom type iso9660 (ro)
/dev/hdd1 on /mnt type ext3 (rw)
```

本章就磁盘分区的创建与管理操作进行了讨论。分区管理是合理规划与分配磁盘存储空间的基础。如果依照本书的进度操作，当前读者的 Linux 操作系统应该仅占用了两个分区，即 root（/）分区与 swap 分区。基于 FHS 的规划，Linux 根目录下的许多目录都可以独立存在于特定的分区，如何对当前的操作系统分区结构进行更合理的分配？如何更方便地管理操作系统分区？这些内容将在下一章展开介绍。

第11章　分区文件系统的管理

对于分区文件系统，在管理上还需要注意以下 3 个问题。

（1）如何实现分区的启动自动挂载。前面介绍了挂载操作，但每次系统重新启动后已经进行的挂载操作都将归于无效，需要重新手动挂载文件。自动挂载操作可帮助用户定义哪些文件系统在启动时要被挂载，并在系统启动过程中自动挂载这些文件系统。

（2）磁盘空间的分配。在一个多用户操作系统中，如何保证每个用户都能按一定比例使用磁盘空间，使磁盘空间不被个别用户或用户群体过多地占用是存储管理的一个重要问题。

（3）如何对 swap 交换分区进行控制，以适应不同的应用需求。

本章就以上 3 个问题展开讨论，分别介绍文件系统的自动挂载、磁盘配额和 swap 控制的内容。本章最后将对文件系统的分区方案进行讨论，以使用户能更好地配置操作系统的分区结构。

11.1　文件系统的自动挂载

操作系统启动后会有哪些文件系统挂载通常是固定的，例如，hda1 挂载在/boot、hda4 挂载在/，hda5 挂载为 swap 等。这些固定的挂载步骤记录在/etc/fstab 文件中，启动系统后自动执行挂载的步骤，也就是会执行 mount -a，而关机时会自动执行卸载。

/etc/fstab 文件是一个文件系统的静态列表文件，用于定义开机后自动挂载的文件系统。可以通过 cat 命令来查看/etc/fstab 文件的内容。

```
[root@xinya /]# cat /etc/fstab
LABEL=/             /           ext3      defaults          1    1
tmpfs               /dev/shm    tmpss     defaults          0    0
devpts              /dev/pts    devpts    gid=5,mode=620    0    0
sysfs               /sys        sysfs     defaults          0    0
proc                /proc       proc      defaults          0    0
LABEL= swap-hda2    swap        swap      defaults          0    0
/dev/hdb1           /home       auto      defaults          1    2
  ↑                   ↑           ↑          ↑              ↑    ↑
 字段 1              字段 2      字段 3      字段 4        字段 5 字段 6
```

该文件被空格分隔为 6 个字段，各字段的含义说明如下。

字段 1-设备名：要挂载的设备，可以使用设备的卷标来表示设备，如 LABEL=/。

字段 2-目录：即挂载点。

字段 3-类型：挂载的文件系统类型。

字段 4-挂载选项：设置挂载时的选项。可以为一个设备的挂载设置多个选项，选项间以“，”分隔。常用的选项见表 11.1。

表 11.1　文件系统的常用挂载选项

选　　项	含　　义
async/sync 异步/同步	设置是否允许磁盘与内存中的数据同步写入。async 为异步写入，sync 为同步写入。使用 async 的方式运行速度会快一些
auto/noauto 自动/非自动	设置启动时是否自动挂载该设备。auto 为自动挂载，noauto 为非自动挂载。若使用 noauto 则在使用 mount -a 时也不会自动挂载
rw/ro 读写/只读	设置该设备是以可读写的方式挂载还是以只读的方式挂载。对于一些 vfat 之类的由其他操作系统控制的分区，可以使用 ro 模式
exec/noexec 可执行/不可执行	设置在该文件系统中是否可进行“执行”操作（即执行该文件系统中的可执行文件）。对于仅保存数据的文件系统而言可以使用 noexec，但相关操作会比较麻烦
user/nouser 用户挂载/用户不能挂载	设置是否允许普通用户使用 mount 命令来挂载该文件系统。一般而言，从安全角度考虑是不允许普通用户使用 mount 命令挂载设备的，所以此处一般使用 nouser
suid/nosuid 允许 SUID 权限/不允许 SUID 权限	设置该文件系统是否允许 SUID 权限的存在。一般而言，如果不是 Linux 系统的分区，而是一般数据分区，则设置 nosuid 是比较安全的，毕竟 SUID 权限对系统而言是存在一定的不安全因素的
defaults 默认值	是一个默认值，该值包含 rw、suid、exec、auto、nouser、async 等参数。如果对待挂载文件系统没有特殊的要求，使用 defaults 即可

字段 5-dump：使用 dump 命令备份的频率。若该字段为 0，系统视为不需要备份；若该字段为 1，表示要进行 dump 备份；若该字段为 2，表示要进行 dump 备份，但要在 1 所标识的文件系统备份完成后再进行。

字段 6-fsck：启动时 fsck 命令会检查文件系统。0 表示不作 fsck 检查；1 表示进行 fsck 检查；2 表示进行 fsck 检查，但要在 1 所标识的文件系统备份完成后再进行。

通过以上对/etc/fstab 文件的介绍，明确了如何在系统启动后自动挂载文件系统。当需要在启动时添加自定义的文件系统时，可以将待挂载的文件系统直接添加至该文件中。

下面以启动时自动加载光驱的应用为实例来介绍/etc/fstab 文件的配置方法。在当前系统中，光驱的设备文件为/dev/hdc。

```
LABEL=/             /             ext3         defaults         1 1
tmpfs               /dev/shm      tmpfs        defaults         0 0
devpts              /dev/pts      devpts       gid=5,mode=620   0 0
sysfs               /sys          sysfs        defaults         0 0
proc                /proc         proc         defaults         0 0
/dev/hdc            /media        iso9660      defaults         0 0
设备名              挂载点        文件系统       选项
```

注意，文件系统类型字段除添加实际的文件系统类型外，还可以使用 auto 命令。该命令允许 Linux 自动探测设备的文件系统类型并挂载。当不清楚被挂载设备的文件系统类型时，使用该命令会更方便。

11.2 磁盘配额——quota

Linux 是一个多用户操作系统，如果系统有很多用户，就必须限制每个用户的保存空间，以免其中某些用户存放了太多文件数据，占用了过多的磁盘容量。quota（配额）就是用来管理用户空间的，也就是设置每个用户主目录的容量限制。

11.2.1 打开 quota 功能

首先要打开含有用户主目录的文件系统的 quota 功能。如果用户为/home 设置单独的硬盘分区用来存放用户主目录，就要打开/home 所在分区的 quota；如果主目录/home 在根分区/里面，就要打开根分区/的 quota。

打开一个文件系统（分区）的 quota 要在/etc/fstab 文件内设置。用 vi 编辑器打开/etc/fstab。下面是一个/etc/fstab 文件的内容：

```
/dev/VolGroup00/LogVol00  /            ext3    defaults        1 1
LABEL=/boot                /boot        ext3    defaults        1 2
tmpfs                      /dev/shm     tmpfs   defaults        0 0
devpts                     /dev/pts     devpts  gid=5,mode=620  0 0
sysfs                      /sys         sysfs   defaults        0 0
proc                       /proc        proc    defaults        0 0
/dev/VolGroup00/LogVol01   swap         swap    defaults        0 0
/dev/hdb1             /home             auto    defaults        1 2
/dev/hdc                /media          iso9660 defaults        0 0
```

现在要打开/home 分区的 quota 功能，应在**/dev/hdb** 的挂载选项上添加 quota 选项：

```
/dev/hdb1   /home     auto    defaults,usrquota,grpquota  1  2
```

grpquota 为打开组 quota 功能，usrquota 则是打开用户 quota 功能。

11.2.2 产生 quota 文件

使用 quota 时需要 quota.user 和 quota.group 两个配置文件，这两个配置文件是系统自动产生的。执行 quotacheck 命令会开始检查要管理的文件系统，然后创建这两个文件。

```
[root@xinya /]# quotacheck -guvam
quotacheck :Scanning /dev/sda1 [/] done
quotacheck :Checked  139596 directories and 151916 files
```

检查完文件系统之后，quotacheck 会自动在文件系统分区的根目录下生成 aquota.group 和 aquota.user 两个控制文件。aquota.group 是用于控制组账号的磁盘配额的，

aquota.user 是用于控制用户账号的磁盘配额的，这两个文件都要用到。

11.2.3　设置 quota

接着就要实际设置管理磁盘空间配额，使用 edquota 来编辑用户或组的 quota 设置。要开始设置 quota 就要先了解分配磁盘空间的方法，也就是对每个用户或组限制其使用容量，限制分为 soft limit 与 hard limit。

（1）hard limit。是分配给每个用户的最大空间。例如，给用户分配 5MB 空间，如果用户超过这个限制就不能再保存文件。

（2）soft limit。用户可以使用超过 soft limit 的空间，但这时系统会对用户发出警告，要求用户清理自己的文件以释放空间，而且若用户在限期（Grace Period）内没有释放出空间，将不能再保存任何文件。

设置用户 quota 与设置组 quota 大致相同，下面介绍以用户为主的 quota 设置。

（1）设置用户 quota。

```
edquota  -u  用户名
```

选项-u 可以不加。例如，设置用户 user1 的 quota，输入：

```
edquota  user1
```

edquota 会启动 vi 编辑器来编辑用户 user1 的 quota 数据。

```
[root@xinya /]# edquota user1
Disk quotas for user user1 (uid 500):
Filesystem    blocks   soft    hard   inodes   soft   hard
/dev/hdb1         44      0       0       11      0      0
```

编辑器中包含 3 个字段。

① Filesystem：表示要在哪个文件系统（分区）上实现磁盘配额。

② blocks：表示该用户已经使用的 block 数，block 的数目可以表示容量的大小。其中：

soft：表示分配给用户的磁盘容量的 soft limit 值。默认为 0，表示不设置。

hard：表示分配给用户的磁盘容量的 hard limit 值。默认为 0，表示不设置。

③ inodes：表示该用户已使用的索引节点数。其中：

soft：表示分配给用户的索引节点的 soft limit 值。默认为 0，表示不设置。

hard：表示分配给用户的索引节点的 hard limit 值。默认为 0，表示不设置。

可以使用编辑功能来修改各个值，以满足用户的需要。如限制 user1 使用 5MB 的磁盘空间，soft limit 为 4.5MB；同时限制 user1 可以建立 100 个文件，soft limit 为 90 个文件。

```
Disk quotas for user user1 (uid 500):
Filesystem        blocks       soft       hard      inodes      soft      hard
/dev/hdb1             44       4608       5120          11        90       100
```

要使用 vi 命令保存退出，则执行以下命令：

```
:wq
```

（2）复制 quota 的设置给其他用户。

设置好一个用户的 quota 配额后，可以把相同的设置复制给其他用户，方法为：

edquota -p 来源 目标

例如，要将 user1 用户的 quota 配额复制给 user2：

```
edquota -p user1 user2
```

而如果要把 user1 用户的配额复制给所有用户，则执行：

```
edquota -p user1
```

设置组 quota 与设置用户 quota 类似，edquota 命令需要使用-g 选项以设置组 quota。

edquota -g 组名称

设置组的 quota 时，最好能够估计一下组内的用户数目及每个用户的 quota 设置，以免二者冲突。

另外，组的 quota 设置也可以使用复制功能，方法如下：

edquota -gp 源组 目标组

设置完 quota 后，可以设置 soft limit 的期限，命令如下：

edquota -t

设置组的期限的命令如下：

edquota -gt

edquota -t 命令同样启动 vi 编辑器进行编辑：

```
Grace period before enforcing soft limits for users:
Time units may be: days, hours, minutes, or seconds
 Filesystem            Block grace period     Inode grace period
 /dev/hdb1                          7days                  7days
```

第一行告知用户限期在 soft limit 值临界时启用。默认限期值为 7 天，也可以修改为 24 小时等格式的时间。更改后的限期会在用户再次超过 soft limit 之后起作用。

11.2.4 执行 quota

设置完成后，必须重新开机，再执行 quota。执行 quota 功能使用 quotaon 命令。

```
quotaon -guva
```

通常需要一启动就自动执行 quota，用户可以在/etc/rc.d/rc.local 文件中加入下面的内容：

```
quotaon  -guva
```

这样每次启动计算机就会自动执行 quota。

如果要关闭 quota，输入：

```
quotaoff  -guva
```

11.2.5 查看 quota

使用 quota 指令检查某一用户 quota 的状态与空间使用情况：

```
quota  [用户]
```

不加用户就显示自己的 quota 情况。

查看系统中所有用户的 quota 情况，输入：

```
quota  -a
```

11.3 swap 管理

虚拟内存是实际物理内存的扩充，当实际的物理内存空间耗用殆尽时，将启用虚拟内存转储物理内存中的数据，以便为物理内存创造更多的可使用空间。

Linux 中虚拟内存有两种表现形式。

（1）分区形式的虚拟内存，即 swap 分区。

（2）文件形式的虚拟内存，即 swap 文件。

这两种形式的虚拟内存形式各有利弊。分区形式的虚拟内存具有速度快、效率高的特点；而文件形式的虚拟内存具有使用灵活、配置简便的特点。

swap 可以在安装操作系统时定义，也可以在操作系统安装完成后定义。在 Linux 中单个 swap 的容量最大为 2GB，系统默认支持 32 个 swap 设备。

本节将介绍两种形式的 swap 的创建和管理方法。

11.3.1 建立分区形式的虚拟内存

在安装操作系统时所定义的 swap 是一种分区形式的 swap。在系统应用过程中，若出现 swap 不足的情况，可以在系统中建立 swap 分区，并将该分区纳入 swap 运行容量，以弥补 swap 的不足。

创建分区型 swap 的方法如下。

（1）利用“fdisk　设备文件名”创建 swap 分区。

swap 分区的容量应为实际物理内存的 1.5~2 倍，文件系统类型代码为 82（swap）。

【示例】在/dev/hdd 设备中创建一个逻辑分区，容量为 2GB，作为 swap 分区使用。

```
[root@xinya ~]# fdisk /dev/hdd
…
Command (m for help): p                           ←查看磁盘分区状态
…
   Device Boot      Start        End      Blocks    Id  System
/dev/hdd1               1       4135     1953756   83  Linux
/dev/hdd2            4136      17753     6434505    5  Extended
/dev/hdd5            4136       8270     1953756   83  Linux
/dev/hdd6            8271      12405     1953756   83  Linux

Command (m for help): n                           ←新建磁盘分区
Command action
   l   logical (5 or over)
   p   primary partition (1-4)
l                                                 ←新建逻辑分区
First cylinder (12406-17753, default 12406):12406
Using default value 12406                         ←容量为 2GB
Last cylinder or +size or +sizeM or +sizeK (12406-17753, default 17753):
+2GB
Command (m for help): t                           ←设置分区类型
Partition number (1-7): 7
Hex code (type L to list codes): 82               ←分区类型编号为 82
Changed system type of partition 7 to 82 (Linux swap / Solaris)
Command (m for help): p
…
   Device  Boot  Start      End     Blocks   Id  System
/dev/hdd1            1     4135    1953756   83  Linux
/dev/hdd2         4136    17753    6434505    5  Extended
/dev/hdd5         4136     8270    1953756   83  Linux
/dev/hdd6         8271    12405    1953756   83  Linux
/dev/hdd7         2406    16540    1953756   82  Linux swap / Solaris
                                                  ←swap 分区

Command (m for help): w                           ←保存退出
```

（2）格式化 swap 分区。

格式化 swap 分区的方法是使用 mkfs 命令。

```
#mkfs  -t  swap 分区设备文件名
```

或

#mkswap　swap 分区设备文件名

```
[root@xinya ~]# mkswap /dev/hdd7
Setting up swapspace version 1, size = 2000642 KB
```

（3）启动和关闭 swap 分区。

启动 swap 分区用如下命令：

#swapon　设备文件名

例如：

```
#swapon  /dev/hdd7
```

关闭 swap 分区用如下命令：

#swapoff　设备文件名

例如：

```
# swapoff  /dev/hdd7
```

11.3.2　建立文件形式的虚拟内存

除分区型 swap 外，文件型 swap 也是 swap 的一种表现形式。文件型 swap 由于是以文件作为存储介质，与分区 swap 相比具有效率低、读写速度慢的缺点。但因文件的配置灵活，不会对分区表造成影响，因此文件型 swap 的灵活性使得其在一些临时应用场合被派上了用场。

创建文件型 swap 的方法如下。

（1）建立 swap 文件。

建立的 swap 文件是一个有特定容量的空文件，其建立的方法是使用 dd 命令。

dd 命令的格式为：

dd　if=源文件　of=目标文件　bs=单位容量　count=单位数量

下面利用 dd 命令创建一个 512MB 的 swap 文件：

```
[root@xinya ~]# swapon /dev/hdd7
[root@xinya ~]# dd if=/dev/zero of=/tmp/swap1 bs=512MB count=1
1+0 records in
1+0 records out
512000000 bytes (512 MB) copied, 4.91006 seconds, 104 MB/s
```

（2）格式化 swap 文件。

使用 mkswap 命令格式化 swap 文件：

```
[root@xinya ~]# mkswap /tmp/swap1
Setting up swapspace version 1, size = 511995 KB
```

（3）启动和关闭 swap 文件。

启动 swap 文件的命令格式如下：

```
swapon  swap 文件名
```

例如：

```
#swapon   /tmp/swap1
```

关闭 swap 文件的命令格式如下：

```
swapoff  swap 文件名
```

例如：

```
#swapoff  /tmp/swap1
```

（4）查看 swap 的使用情况。

查看 swap 的使用情况可以使用 free 命令。

```
[root@xinya ~]# free
              total       used        free      shared     buffers       cached
Mem:        255412      249964       5448          0       980           114360
-/+ buffers/cache:       134624      120788
Swap:       2978020       140      2977880
```

11.4 主机分区与目录配置

在掌握了磁盘分区和磁盘挂载方法之后，可以对当前系统的目录和其所在的分区进行优化，以提高安全性和便利性。

对于 Linux 主机而言，并不建议将所有的目录统一放置在一个分区中，其原因可以从两个方面来说明。

（1）安全性考虑。将所有目录放置在一个分区中，当分区或系统被破坏时，将导致所有目录下的数据被连带地破坏。所以，应尽量将数据目录与系统功能性目录分别放置在不同的分区。

（2）便利性。如果需要升级系统时，仅需要将相关的目录数据更新即可，由于目录存放在不同的分区中，这时可以很方便地通过对相关分区的卸载将不相关数据进行分离。

Linux 的分区方案需要依据各自系统的需要而定，系统中可以独立存在于一个分区中的目录如下：

（1）/（该分区中必须要包含/etc、/sbin、/bin、/dev 和/lib 这几个基本目录）

（2）/home

（3）/boot

（4）/var

（5）/usr

（6）/tmp

至此，用户就可以根据前面介绍的分区与挂载的相关知识将各个可单独存储的目录移至独立的磁盘分区中，并挂载到根目录下使用了。

本章讨论了分区的管理与规划。其中，主机分区与目录配置并未强调配置方法，因为前面的章节已经对具体的配置方法进行了介绍。读者可依据前已述及的知识点试着独立完成分区的迁移工作。

第12章　文件系统的归档管理

文件系统的归档管理主要讨论文件的打包、压缩与备份等一系列主题。文件系统的归档管理在 GUI 图形界面下的操作是直观而简洁的，但是在命令行环境下则需要用户控制各种归档程序来完成相关的操作。由于归档程序数量较多，且应用灵活，因此读者有必要系统地了解相关的知识，以便更好地驾驭各种归档程序的应用。

12.1　文件的打包与压缩

为了方便对系统内的文件进行整理、备份以及通过网络传输，最好能将文件打包为一个单独的文件，同时进行压缩以节省空间。大部分 Linux 系统下的应用程序会放在网络上供用户下载，有很多会是压缩文件的形式，这时用户就需要学会如何解压缩文件了。本节分别说明常见的 tar、zip 和 gzip 文件的压缩与解压缩。

12.1.1　磁带文件

早期计算机的存储设备是磁带机，tar 就是管理磁带文件（tape archive）的程序。tar 可以用于打包文件，即用户可以把包含子目录的多个文件使用 tar 打包成一个文件，不过 tar 没有压缩文件的功能，必须配合 gzip 或 compress 使用才能实现对目标文件的压缩。下面就介绍 tar 的用法。

1. 将多个文件打包为一个文件

将多个文件打包为一个文件的命令格式如下：

```
tar  -cvf 目标文件.tar file1 … filen
```

选项说明如下：

-c：建立打包文件。

-v：显示执行过程。

-f：使用指定的归档文件。注意，f 后要直接加目标文件名，不要再添加选项。

目标文件.tar：即打包的 tar 文件。

file1 … file*n*：为要打包的文件，也可以是目录，tar 会把目录下的文件都打包。

-cvf 选项会创建一个新的.tar 文件，然后将文件 file1 到 file*n* 加入打包文件，并显示操作过程。

【示例】将文件 file1、file2、file3、file4 和目录 filedir 打包为 file.tar 文件。

```
[root@xinya ~]# ll
…
-rw-r--r--  1  root  root 112640  05-14  18:28  file1
-rw-r--r--  1  root  root  33726  05-14  18:27  file2
-rw-r--r--  1  root  root   7117  05-14  18:27  file3
-rw-r--r--  1  root  root  59364  05-14  18:27  file4
drwxr-xr-x  2  root  root   4096  05-14  18:26  filedir
[root@xinya ~]# tar  -cvf  file.tar  file1  file2  file3  file4  filedir
file1
file2
file3
file4
filedir/
[root@xinya ~]# ll
…
-rw-r--r--  1  root  root 112640  05-14  18:28  file1
-rw-r--r--  1  root  root  33726  05-14  18:27  file2
-rw-r--r--  1  root  root   7117  05-14  18:27  file3
-rw-r--r--  1  root  root  59364  05-14  18:27  file4
drwxr-xr-x  2  root  root   4096  05-14  18:26  filedir
-rw-r--r--  1  root  root 225280  05-14  18:28  file.tar
                                   ←file.tar 文件的容量是各文件和目录的总和
```

2. 显示.tar 的内容

要知道一个 tar 文件中包含哪些内容，方式是：

tar -tvf 文件名.tar

选项说明如下：

-t：查看文件包中的内容。

-v：显示命令的执行过程，在这里将显示文件包中文件的详细信息。如果不加该选项，则只显示文件名信息。

-f：使用指定的归档文件。

【示例】查看 file.tar 包文件中包含哪些文件。

```
[root@xinya ~]# tar -tvf file.tar
-rw-r--r-- root/root   112640  2011-05-14  18:28:03  file1
-rw-r--r-- root/root    33726  2011-05-14  18:27:20  file2
-rw-r--r-- root/root     7117  2011-05-14  18:27:37  file3
-rw-r--r-- root/root    59364  2011-05-14  18:27:46  file4
drwxr-xr-x root/root        0  2011-05-14  18:26:29  filedir/
```

3. 将文件添加到 tar 文件内

要在已打包的.tar 文件中再加入文件，需要使用-r 选项。

tar -rvf 文件名.tar file1 … filen

选项说明如下：

-r：向指定的包文件中附加文件。

-r 选项会把 file1 … file*n* 加入到已存在的.tar 文件中。

【示例】将 test1、test2 和 est3 添加到 file.tar 文件中。

```
[root@xinya ~]# ll
drwxr-xr-x  3  root  root     4096   2011-12-31    Desktop
-rw-r--r--  1  root  root   112640   05-14  18:28  file1
-rw-r--r--  1  root  root    33726   05-14  18:27  file2
-rw-r--r--  1  root  root     7117   05-14  18:27  file3
-rw-r--r--  1  root  root    59364   05-14  18:27  file4
drwxr-xr-x  2  root  root     4096   05-14  18:26  filedir
-rw-r--r--  1  root  root   225280   05-14  18:28  file.tar
-rw-r--r--  1  root  root  2097152   05-14  18:31  test1
-rw-r--r--  1  root  root  1048576   05-14  18:31  test2
-rw-r--r--  1  root  root  1048576   05-14  18:31  test3
[root@xinya ~]# tar -rvf file.tar test1 test2 test3  ←将文件加入打包文件
test1
test2
test3
[root@xinya ~]# tar -tvf file.tar
-rw-r--r--  root/root   112640  2011-05-14  18:28:03  file1
-rw-r--r--  root/root    33726  2011-05-14  18:27:20  file2
-rw-r--r--  root/root     7117  2011-05-14  18:27:37  file3
-rw-r--r--  root/root    59364  2011-05-14  18:27:46  file4
drwxr-xr-x  root/root        0  2011-05-14  18:26:29  filedir/
-rw-r--r--  root/root  2097152  2011-05-14  18:31:35  test1
-rw-r--r--  root/root  1048576  2011-05-14  18:31:44  test2
                                                  ←文件已被加入到打包文件
-rw-r--r--  root/root  1048576  2011-05-14  18:31:47  test3
```

4. 将文件由 tar 内删除

如果在.tar 文件中有一个或几个文件要从.tar 文件中删除，可以使用--delete 选项。

tar --delete -vf 文件.tar file1 … filen

选项说明如下：

--delete：由归档文件中删除文件。

--delete 选项会将文件 file1 到 file*n* 从文件.tar 中删除。

【示例】将 test1、test2 和 test3 由 file.tar 文件中删除。

```
[root@xinya ~]# tar -tvf file.tar
-rw-r--r--  root/root   112640  2011-05-14  18:28:03  file1
-rw-r--r--  root/root    33726  2011-05-14  18:27:20  file2
-rw-r--r--  root/root     7117  2011-05-14  18:27:37  file3
-rw-r--r--  root/root    59364  2011-05-14  18:27:46  file4
drwxr-xr-x  root/root        0  2011-05-14  18:26:29  filedir/
-rw-r--r--  root/root  2097152  2011-05-14  18:31:35  test1
-rw-r--r--  root/root  1048576  2011-05-14  18:31:44  test2
-rw-r--r--  root/root  1048576  2011-05-14  18:31:47  test3
[root@xinya ~]# tar --delete -vf file.tar test1 test2 test3
[root@xinya ~]# tar -tvf file.tar
-rw-r--r--  root/root   112640  2011-05-14  18:28:03  file1
-rw-r--r--  root/root    33726  2011-05-14  18:27:20  file2
-rw-r--r--  root/root     7117  2011-05-14  18:27:37  file3
-rw-r--r--  root/root    59364  2011-05-14  18:27:46  file4
drwxr-xr-x  root/root        0  2011-05-14  18:26:29  filedir/
```

5. 解开.tar 包文件

解开.tar 包文件的方法为：

tar -xvf 文件.tar

选项说明如下：

-x：解开打包文件。

【示例】将 file.tar 文件移动到/tmp 目录下并解包。

```
[root@xinya tmp]# ls
file.tar
[root@xinya tmp]# clear
[root@xinya tmp]# tar -xvf /tmp/file.tar
file1
file2
file3
file4
filedir/
[root@xinya tmp]# ls
file1  file2  file3  file4  filedir  file.tar
```

这里需要注意的是，tar 将文件解包到当前工作目录下。

下面对 tar 命令的用法加以说明。

【语法】**tar --delete -ctrxvf [文件.tar] [file1 … file*n*]**

选项说明如下：

-c：建立打包文件。

-v：显示执行过程。

-f：使用指定的归档文件。注意，f 后要直接加目标文件名，不要再添加选项。

-t：查看文件包中的内容。

-r：向指定的包文件中附加文件。

-x：解开打包文件。

--delete：由包文件中删除文件。

12.1.2　利用 compress/uncompress 压缩和解压缩文件

compress 命令是 UNIX 上传统的压缩文件的命令，解压缩则使用 uncompress 命令。由于 compress 程序的压缩效果不是很好，现在已经很少使用。

本书所采用的 CentOS 5.x 发行版默认未安装 compress 程序，读者可以手工安装该程序。

compress 压缩程序的安装包位于安装光盘的 CentOS 目录下，软件包名称为 ncompress-4.2.4-47.i386.rpm。将安装挂载到/mnt 目录下，进入 CentOS 目录下，将 ncompress-4.2.4-47.i386.rpm 软件包复制到/usr/local 目录下进行安装。安装该软件包时使用“rpm -ivh ncompress-4.2.4-47.i386.rpm”命令。

```
[root@xinya ~]# mount /dev/cdrom /mnt
mount: block device /dev/cdrom is write-protected, mounting read-only
[root@xinya ~]# cd /mnt/CentOS/
[root@xinya CentOS]# ls ncompress-4.2.4-47.i386.rpm
ncompress-4.2.4-47.i386.rpm
[root@xinya CentOS]# cp ncompress-4.2.4-47.i386.rpm /usr/local
[root@xinya CentOS]# cd /usr/local
[root@xinya local]# ls
bin  games    lib      ncompress-4.2.4-47.i386.rpm  share
etc  include  libexec  sbin                         src
[root@xinya local]# rpm -ivh ncompress-4.2.4-47.i386.rpm
warning: ncompress-4.2.4-47.i386.rpm: Header V3 DSA signature: NOKEY, key
ID e8562897
Preparing...          ########################################### [100%]
   1:ncompress          ########################################### [100%]
```

安装完成后即可使用 compress 程序了。

1. 压缩文件

利用 compress 程序压缩文件，压缩后会自动为文件添加“.Z”扩展名，且压缩后的文件会取代源文件。

压缩文件的命令格式如下：

```
compress  文件名
```

【示例】利用 compress 程序对/tmp 目录下的 file.tar 文件进行压缩。

```
[root@xinya tmp]# ll
……
-rw-r--r-- 1 root root 225280 05-14 19:04 file.tar
……
[root@xinya tmp]# compress  file.tar
[root@xinya tmp]# ll
……
-rw-r--r-- 1 root root  68537 05-14 19:04 file.tar.Z
……
```

在利用 compress 压缩文件时，还可以利用-b 选项来定义压缩等级。

```
compress  -bn  文件
```

-b 选项后定义压缩等级 *n*，*n* 为 9~16 的数字，表示压缩的程度，16 为最高，压缩比率最大，运行时间也最长。压缩过的文件会自动添加扩展名.Z。

2. 解压缩

解压缩的方法如下：

```
uncompress  文件.Z
```

解压缩后的文件会取代压缩文件.Z。

注意，由于 compress 命令的压缩效果较差，所以现在已经很少使用。在一些 Linux 发行版中已经不再提供该命令了。

12.1.3 利用 zip/unzip 压缩和解压缩文件

zip 是一个使用很广的压缩方法，在很多其他操作系统中也可以见到使用 zip 程序压缩的文件。zip 支持对多文件压缩，如果有许多文件需要处理，使用 zip 是一个很好的处理方法。

1. 压缩文件

使用 zip 压缩文件的基本方法如下：

```
zip  zipfile  file1 … filen
```

zip 会将 file1 到 file*n* 等文件压缩成 zipfile.zip（注意文件的扩展名.zip 是由压缩程序自动添加的，不需要用户添加）。当需要把其他文件加入到已存在的 zipfile.zip 文件中时也是使用相同的语法，如果相同文件名已经存在于 zipfile.zip 中，就会用新加入的文件覆盖同名原文件，不存在的文件则会被新增。

【示例】使用 zip 程序将 file1、file2 和 file3 三个文件压缩为 myfile.zip 文件。

```
[root@xinya ~]# ll -h
-rw-r--r-- 1  root  root  110K   05-14  18:28  file1
-rw-r--r-- 1  root  root   33K   05-14  18:27  file2
-rw-r--r-- 1  root  root  7.0K   05-14  18:27  file3
[root@xinya ~]# zip  myfile  file1  file2 file3
  adding: file1 (deflated 84%)
  adding: file2 (deflated 65%)
  adding: file3 (deflated 74%)
[root@xinya ~]# ll -h
-rw-r--r-- 1  root  root  110K   05-14  18:28  file1
-rw-r--r-- 1  root  root   33K   05-14  18:27  file2
-rw-r--r-- 1  root  root  7.0K   05-14  18:27  file3
-rw-r--r-- 1  root  root   31K   05-14  20:23  myfile.zip
```

2. 压缩目录

zip 在处理文件时如果遇到目录名称，只会把目录名称加入到 zip 文件中，目录内的文件则不加入。如果要连同目录下的文件和子目录一并加入压缩，需要使用-r 选项。

zip -r zipfile 文件 1 目录 1 …

3. 不压缩特定的文件

这个方法可以指定特定扩展名的文件加入 zip 文件而不压缩。

zip -n .扩展名 1: .扩展名 2 … zipfile file1 … filen

用户可以指定多个特定的扩展名，以“:”分隔。凡是具有指定扩展名的文件，都以不压缩的形式进入 zip 文件。

4. 特定的文件不加入 zip 文件

用户可以指定不加入 zip 文件的文件。

zip zipfile file1 … filen -x list

list 是一个文件列表，该列表中的文件不加入 zip 文件。另外也可以指定加入的文件：

zip zipfile file1 … filen -i list

只有 list 列表中出现的文件才会加入 zip 文件。

5. 从 zip 文件中删除文件

zip 文件中如果有文件要删除，可以直接使用 zip 命令的-d 选项删除。

```
zip  -d  zipfile.zip  file1 … filen
```

文件 file1 到 file*n* 会被从 zip 文件中删除。

6. 解压缩

解压缩使用 unzip 命令，基本用法如下：

```
unzip  zipfile.zip
```

这样会把 zipfile.zip 的所有文件解压缩。注意，如果文件默认被解压到当前工作目录，且解压后的文件与当前目录中的现有文件重名，程序会询问用户是否以解压缩的文件代替或是重命名文件。

7. 显示 zip 文件内容

unzip 命令可以用于检查 zip 文件的内容：

```
unzip  -v  zipfile.zip
```

```
[root@xinya ~]# unzip -v myfile.zip
Archive:  myfile.zip
Length  Method    Size   Ratio  Date       Time    CRC-32     Name
------  -----     -----  ----   ----       ----    -----      ---
112640  Defl:N    17714  84%    05-14-11  18:28   02b3b4d3   file1
33726   Defl:N    11743  65%    05-14-11  18:27   c8d3b39c   file2
7117    Defl:N     1842  74%    05-14-11  18:27   22a8bb2b   file3
-----             -----  ---                                 ------
53483             31299  80%                                 3 files
```

8. 将文件解压缩到特定目录下

命令格式如下：

```
# unzip  文件名.zip  -d 目录名
```

zip 压缩文件中的内容均会被解压到-d 选项所指定的目录中。

12.1.4　利用 gzip 压缩和解压缩文件

gzip 也是一个压缩程序，压缩后的文件扩展名为.gz。

1. 压缩文件

使用 gzip 压缩文件的基本用法如下：

```
# gzip  文件名
```

gzip 会自动为压缩后的文件添加.gz 的扩展名，并取代原来的文件。如果要显示文件

的压缩比，就要加上-v 选项：

gzip -v 文件名

【示例】将 file1、file2、file3 和 file4 打包成 gfile.tar 文件，并使用 gzip 程序进行压缩。

```
#gzip -v gfile.tar
[root@xinya ~]# gzip -v file1 file2 file3
file1:   84.3% -- replaced with file1.gz
file2:   65.2% -- replaced with file2.gz
file3:   74.1% -- replaced with file3.gz
[root@xinya ~]# ll
-rw-r--r--  1  root  root   17738  05-14  18:28  file1.gz
-rw-r--r--  1  root  root   11767  05-14  18:27  file2.gz
-rw-r--r--  1  root  root    1866  05-14  18:27  file3.gz
```

2. 查看压缩文件中的内容

查看.gz 文件内容的方法如下：

gzip -l 文件名.gz

```
[root@xinya ~]# gzip -l file1.gz
compressed       uncompressed   ratio  uncompressed_name
   17738              112640     84.3%  file1
```

3. 解压缩

gzip 有另一个解压缩命令 gunzip，实际上是 gzip 命令的符号链接。解压缩的方法如下：

gzip -d 文件

-d 选项会将文件.gz 解压缩成文件并取代原文件.gz。

4. gzip 命令与 tar 命令合并使用

gzip 常与 tar 命令联合使用。在使用过程中，可以将打包与压缩的过程或解包与解压缩的过程合并在一个命令中完成，该命令就是 tar 的-z 选项。

将文件打包并使用 gzip 压缩的命令是：

tar -zcvf 目标文件.tar.gz file1 … filen

选项说明如下：

-z：使 tar 同时具有 gzip 属性，即打包之后直接使用 gzip 程序压缩。

将文件使用 gzip 解压缩后使用 tar 命令解包的命令是：

```
# tar  -zxvf 目标文件.tar.gz
```

选项说明如下：

-z：使 tar 同时具有 gzip 属性，即使用 gzip 程序解压缩后使用 tar 命令解包。

12.1.5　利用 bzip2 压缩和解压缩文件

有一类压缩文件的扩展名是.bz2，它是一个新的压缩程序-bzip2 产生的压缩文件。网络上有一些文件的压缩方式会选用 tar 与 bzip2 并用，用户会看到.tar.bz2 的扩展名，通常系统的内核或更新文件会以这个方式压缩。

1．压缩文件

使用 bzip2 压缩文件的基本方法如下：

```
bzip2  文件
```

bzip2 会把文件压缩成文件.bz2 取代原来的文件。如果需要保留原来的文件，就要加上-k 选项：

```
bzip2  -k  文件名
```

2．解压缩

bzip2 有另一个解压缩命令 bunzip2，实际上是 bzip2 命令的符号链接。解压缩的方法如下：

```
bunzip2  文件名
```

或

```
bzip2  -d  文件名
```

3．修复文件

.bz2 的压缩文件如果发生损坏，可以尝试使用 bzip2recover 来修复。修复的方法如下：

```
bzip2recover  文件名
```

4．bzip2 命令与 tar 命令合并使用

由于 bzip2 仅支持对单一文件进行压缩，所以 bzip2 常与 tar 命令联合使用。在使用过程中，可以将打包与压缩的过程或解包与解压缩的过程合并在一个命令中完成，该命令就是 tar 的-j 选项。

将文件打包并使用 bzip2 压缩的命令是：

```
# tar -jcvf 文件名.tar.bz2 file1 … filen
```

选项说明如下：

-j：使 tar 同时具有 bzip2 属性，即打包后直接使用 bzip2 进行压缩。

将文件使用 bzip2 解压缩后使用 tar 命令解包的命令是：

```
# tar -jxvf 目标文件.tar.gz
```

选项说明如下：

-j：使 tar 同时具有 bzip2 属性，即使用 bzip2 程序解压缩后使用 tar 命令解包。

12.2 文件系统的备份

文件系统的备份（backup）是指从原文件中独立出来的、单独存储的程序或文件的副本。备份对于现代文件系统和操作系统而言是十分重要的。

12.2.1 备份概述

在操作系统中备份分为两种类型。

（1）系统备份，是指将计算机操作系统的文件、结构和状态等数据保存起来，当操作系统发生故障时可以从系统备份中恢复操作系统的状态和数据。

（2）数据备份，是指将文件、数据库和应用程序生成副本保存起来，以备数据恢复时使用。数据备份可以直接使用 cp 等复制命令来实现。

12.2.2 备份的方法

对于备份操作而言，当一个文件被备份后，该文件会获得一个“已备份”标识，而当文件属性和内容发生变化时，文件又会获得一个“未备份”标识。当备份时，就是依靠文件的“已备份”和“未备份”标识来确定要备份哪些文件的。

备份采用以下三种方法：

（1）完全备份：是对目标进行完整的备份，不考虑文件之前是否被备份过，所有文件均被标识为“已备份”。

（2）差异备份：是针对上一次完全备份和增量备份而言的，备份上一次备份后发生变化的所有文件。差异备份不会为文件添加“已备份”标识。

（3）增量备份：是针对上次备份（完全备份和增量备份）而言的，备份上一次备份后发生变化的所有文件。增量备份会为文件添加“已备份”标识。

12.2.3 Linux 的备份工具 dump

在 Linux 操作系统中，文件的备份使用 dump 命令来完成。dump 是一个可针对整个文件系统或单一目录进行备份的工具，所支持的备份方法包括完整备份和差异备份两种，备份对象可以使用挂载点或设备文件名来表示。但 dump 对普通文件目录进行备份时，则只能执行完整备份。

1. 查看备份文件系统所需要的磁盘空间

命令格式如下：

dump -S 文件系统名

【示例】查看备份/dev/hda1 文件系统所需的磁盘空间。

```
[root@xinya ~]# dump -S /dev/hda1
6701056
```

2. 完整备份文件系统

备份文件系统时要明确文件系统是完整备份还是差异备份，完整备份用 0 表示，差异备份用 1~9 表示。差异备份的“1”表示在“0”的基础上进行差异备份，“2”表示在“1”的基础上进行差异备份，……，依此类推。

利用 dump 命令完整备份文件系统的语法结构为：

dump -0 文件系统设备名|挂载点目录名

备份时，若需要将备份时间保留下来，可以使用-u 选项，该选项用于将文件系统的备份时间记录到/etc/dumpdates 文件中。

dump -0u 文件系统设备名|挂载点目录名

【示例】对文件系统/dev/hda1 进行备份。

```
[root@xinya ~]# dump  -0u  /dev/hda1
DUMP: Date of this level 0 dump: Sat May 14 21:14:45 2011
DUMP: Dumping /dev/hda1 (/boot) to /dev/tape
                                    ←文件系统默认被备份到/dev/tape 文件中
DUMP: Label: /boot
DUMP: Writing 10 Kilobyte records
DUMP: mapping (Pass I) [regular files]
DUMP: mapping (Pass II) [directories]
DUMP: estimated 6544 blocks.
DUMP: Volume 1 started with block 1 at: Sat May 14 21:14:45 2011
DUMP: dumping (Pass III) [directories]
DUMP: dumping (Pass IV) [regular files]
DUMP: Closing /dev/tape
DUMP: Volume 1 completed at: Sat May 14 21:14:46 2011
DUMP: Volume 1 6600 blocks (6.45MB)
DUMP: Volume 1 took 0:00:01
DUMP: Volume 1 transfer rate: 6600 KB/s
DUMP: 6600 blocks (6.45MB) on 1 volume(s)
DUMP: finished in 1 seconds, throughput 6600 kBytes/sec
```

```
DUMP: Date of this level 0 dump: Sat May 14 21:14:45 2011
DUMP: Date this dump completed:  Sat May 14 21:14:46 2011
DUMP: Average transfer rate: 6600 KB/s
DUMP: DUMP IS DONE
```

若需要指定备份文件的存储路径，需要在 dump 命令中使用-f 选项。

dump -0u -f 备份文件存储路径 文件系统设备名|挂载点目录名

【示例】对文件系统/dev/hda1 进行备份，并将备份文件存储于/tmp/backup1 文件中。

```
[root@xinya ~]# dump  -0u -f /tmp/backup1 /dev/hda1
  DUMP: Date of this level 0 dump: Sat May 14 21:15:17 2011
  DUMP: Dumping /dev/hda1 (/boot) to /tmp/backup1  ←备份文件的存储位置
  DUMP: Label: /boot
  DUMP: Writing 10 Kilobyte records
  DUMP: mapping (Pass I) [regular files]
  DUMP: mapping (Pass II) [directories]
  DUMP: estimated 6544 blocks.
  DUMP: Volume 1 started with block 1 at: Sat May 14 21:15:17 2011
  DUMP: dumping (Pass III) [directories]
  DUMP: dumping (Pass IV) [regular files]
  DUMP: Closing /tmp/backup1
  DUMP: Volume 1 completed at: Sat May 14 21:15:18 2011
  DUMP: Volume 1 6600 blocks (6.45MB)
  DUMP: Volume 1 took 0:00:01
  DUMP: Volume 1 transfer rate: 6600 KB/s
  DUMP: 6600 blocks (6.45MB) on 1 volume(s)
  DUMP: finished in 1 seconds, throughput 6600 KB/s
  DUMP: Date of this level 0 dump: Sat May 14 21:15:17 2011
  DUMP: Date this dump completed:  Sat May 14 21:15:18 2011
  DUMP: Average transfer rate: 6600 KB/s
  DUMP: DUMP IS DONE
```

要查看/etc/fstab 文件中记录的分区是否备份过，可以使用 dump 的-w 选项。

```
[root@xinya ~]# dump -w
Dump these file systems:
  /dev/mapper/VolGroup00-LogVol00       (     /) Last dump: never
```

下面对 dump 命令的用法说明如下。

【语法】**dump** [选项] [备份方法] [-f 备份文件名] 文件系统设备名|挂载点目录名

选项说明如下：

-S：列出待备份目标所要占用的磁盘空间。

-u：将此备份的时间记录到/etc/dumpdatas 文件中。

-v：显示命令的执行过程。

-j：与 bzip2 压缩命令联合使用，对备份文件进行压缩。

-f：使用指定的归档文件，定义备份文件的存储位置和文件名。

-w：显示/etc/fstab 文件中记录的分区的 dump 备份状态。

[备份方法]：用 0~9 表示，完整备份用 0 表示，差异备份使用 1~9 表示。差异备份的 1 表示在 0 的基础上进行差异备份，2 表示在 1 的基础上进行差异备份，依此类推。

当利用 dump 命令备份目录时，需要注意的是，dump 只支持对目录进行完整备份，同时不支持将备份时间写入/etc/dumpdataes 文件中。

12.2.4 备份的还原

对于备份文件而言，其还原操作是在恢复故障时的必要操作。备份的还原操作有两种模式可供选择，一种是将整个文件系统还原，此时，需要为待还原的文件系统备份创建一个新的分区，将备份还原至该分区（不支持覆盖还原）。另一种方式是将文件系统中的部分内容进行还原，这需要借助于交互式还原。

Linux 系统中对备份文件的还原可以使用 restore 命令。

1. 利用 restore 命令查看 dump 备份文件

命令格式如下：

```
restore  -t  -f  备份文件名
```

【示例】查看/tmp/backup1 备份文件中的内容。

```
[root@xinya tmp]# restore  -t -f backup1
Dump   date: Sat May 14 21:15:17 2011
Dumped from: the epoch
Level 0 dump of /boot on xinya:/dev/hda1
Label: /boot
         2         .
        11         ./lost+found
     16065         ./grub
     16083         ./grub/grub.conf
     16066         ./grub/splash.xpm.gz
…
```

2. 利用 restore 命令比较备份文件与当前文件系统间的差异

命令格式如下：

```
# restore  -C  -f  备份文件名
```

【示例】比较/tmp/backup1 与当前文件系统间的差异。

```
[root@xinya tmp]# restore -C -f /tmp/backup1
Dump   date: Sat May 14 21:15:17 2011
Dumped from: the epoch
Level 0 dump of /boot on xinya:/dev/hda1
Label: /boot
filesys = /boot
.: mode changed from 0755 to 0777.  ←当前文件系统的权限发生了变化
Some files were modified!  1 compare errors
```

3. 还原整个文件系统

在还原整个文件系统时要注意，由于 restore 命令不支持覆盖式还原，所以应将被还原文件系统还原至其他分区。

命令格式如下：

restore -r -f 备份文件名

注意，备份的数据将还原至当前目录下。

还原时要注意还原顺序，要先还原 0 级备份，然后依次是 1、2、3、…等级别的备份。

4. 交互式还原特定的文件或目录

restore 命令提供了一个交互式应用环境，用于还原特定的文件或目录。

计入交互模式可以使用 restore 命令的-i 选项。

restore -i -f 备份文件名

```
[root@xinya tmp]# restore -i -f backup1
restore > help     ←查看 restore 交互式环境的帮助信息
Available commands are:
  ls [arg] - list directory
  cd arg - change directory
  pwd - print current directory
  add [arg] - add 'arg' to list of files to be extracted
                                        ←将文件和目录加入还原列表
  delete [arg] - delete 'arg' from list of files to be extracted
                                        ←由还原列表中删除
  extract - extract requested files     ←还原还原列表中的文件
  setmodes - set modes of requested directories
  quit - immediately exit program       ←退出 restore 程序
  what - list dump header information
  verbose - toggle verbose flag (useful with "ls")
  prompt - toggle the prompt display
  help or '?' - print this list
If no 'arg' is supplied, the current directory is used
```

12.3 备份相关工具

12.3.1 将备份数据刻录至光盘

备份数据可以存储于磁盘、磁带机和磁盘阵列中，也可以刻录到光盘中保存。Linux 系统中的刻录工具被称为 cdrecord，下面介绍该程序的使用方法。

1. 查询刻录机的位置

命令格式如下：

```
cdrecord  -scanbus  dev=ATA
```

```
[root@xinya tmp]# cdrecord  -scanbus dev=ATA
Cdrecord-Clone 2.01 (cpu-pc-linux-gnu) Copyright (C) 1995-2004 J. Schilling
Note: This version is an unofficial (modified) version with DVD support
Note: and therefore may have bugs that are not present in the original.
Note: Please send bug reports or support requests to http://bugzilla.
redhat.com/ bugzilla
Note: The author of cdrecord should not be bothered with problems in this
version.
scsidev: 'ATA'
devname: 'ATA'
scsibus: -2 target: -2 lun: -2
Linux sg driver version: 3.5.27
Using libscg version 'schily-0.8'.
cdrecord: Warning: using inofficial libscg transport code version (schily
- Red   Hat-scsi-linux-sg.c-1.83-RH '@(#)scsi-linux-sg.c   1.83 04/05/20
Copyright  1997 J. Schilling').
scsibus1:
    1,0,0  100) 'NEC' ' IDE CDR10' '1.00' Removable CD-ROM ←刻录机位置 1,0,0
    1,1,0   101) *
    1,2,0   102) *
    1,3,0   103) *
    1,4,0   104) *
    1,5,0   105) *
    1,6,0   106) *
    1,7,0   107) *
```

2. 擦除光盘内容

若用户使用的是可擦写刻录光盘，可以使用 cdrecord 程序将光盘已有内容擦除。

```
cdrecord  -v  dev=ATA:1,0,0  blank=fsat
```

3. 刻录 CD

若使用 CD 光盘刻录，则 cdrecord 的命令为：

```
cdrecord -v dev=ATA:1,0,0 fs=8m -dummy -data 镜像文件名
```

4. 刻录 DVD

若使用 DVD 光盘刻录，则 cdrecord 的命令为：

```
cdrecord -v dev=ATA:1,0,0 fs=8m -data -sao driveropts=burnfree 镜像文件名
```

12.3.2 文件复制工具 dd

dd 命令可读取文件或设备内容，并将读取到的内容保存为一个文件。因此利用 dd 命令可以实现对设备内容的备份。命令格式如下：

```
dd if=设备文件名 of=备份文件名 bs=单位容量 count=总数
```

【示例】备份硬盘 MBR（第一扇区）的内容。

```
[root@xinya tmp]# dd if=/dev/hda of=/tmp/mbr bs=512 count=1
1+0 records in
1+0 records out
512 bytes (512 B) copied, 6.4859e-05 seconds, 7.9 MB/s
```

注意，bs 的默认值为 512。

本章就文件系统的归档管理进行了讨论，包括文件的打包操作、压缩操作以及备份操作。归档管理是文档整理和传送的有效手段，同时在文档的版本控制方面也具有重要的意义。

本章重点是 tar 程序与其他压缩程序的联合使用。这一点在以后的操作系统扩充中还会不断地被提及，这是因为需要利用该方法来处理相关的源代码软件包。

第13章　软件系统扩充

操作系统在使用的过程中需要添加一系列的应用软件来扩充系统功能，完善运行环境，以期发挥最佳的系统应用效率。

在 Linux 操作系统中，软件的添加与删除操作的形式是多样的，不同的 Linux 发行版采用了不同的软件配置机制。本章就 Linux 操作系统中应用程序软件的添加与删除操作进行介绍，主要涉及源代码软件的安装以及 RPM 包管理器操作等内容。

在当前的 Linux 操作系统中，软件的安装方式从总体上可分为两类：

（1）利用应用程序的源代码进行安装。

（2）利用可执行程序进行安装。

这两种安装方式各有其特点。

利用应用程序的源代码进行安装的方式具有适应性广泛，受 Linux 操作系统的发行版本、内核版本以及应用平台的限制程度低等优点。但同时也具有安装复杂、对系统相关资源的要求程度高等不足。这种不足同时也成为 Linux 灵活性的一种表现，所以此处讲的不足是相对于“利用可执行程序进行安装”这种方式而言的。只是对于初学者而言，这种安装方式不够友好，但这种安装方式却是最根本、最有效的一种安装方式。

利用可执行程序进行安装是另外一种安装方式。这种安装方式强调软件要安装在特定内核版本的特定 Linux 发行版中。虽然安装过程简单，但对系统的选择性比较强。特别是在当前 Linux 各种发行版层出不穷的情况下，过分强调内核版本与发行版特性会导致软件的应用范围过小，不利于应用程序的普及。

如今，Linux 环境下的软件开发者已经意识到这个问题。所以，一般 Linux 环境下的软件在发布时除按 GPL 授权公布软件的源代码外，还会发布针对不同发行版或管理机制的可执行安装程序，如 rpm 格式的软件安装包、dpkg 格式的软件安装包等。

本章首先介绍如何利用应用程序的源代码进行软件安装。

13.1　应用程序的源代码安装方式

利用应用程序的源代码进行软件安装，要求用户有一定的软件编译知识。不过，为尽量减少这种对系统用户开发知识的依赖，当前系统中提供了一系列机制来提高源代码软件编译与安装的友好程度。尽管如此，我们还是要对软件的源代码编译做一个介绍。

由于 Linux 操作系统中的应用程序大多采用 C 语言进行开发，因此本章所涉及的软

件均是采用 C 语言以及 C 语言编译器进行开发和编译的。

软件的开发，或者说软件的产生，是一个系统工程。其基本流程如下。

（1）利用高级语言（C 语言）编写应用程序源代码，获得应用程序的源代码文件。应用程序的源代码文件往往以“.c”作为其扩展名。注意，此时的应用程序源文件只有用户可读，因为操作系统内核不能识别高级语言所表达的内容。

（2）利用 C 语言编译器（常用的是 gcc 或 cc）对源代码文件进行编译，将由高级语言编写的源程序编译成操作系统内核能够识别的、以二进制代码表示的目标程序。目标程序文件往往以“.o”作为其扩展名。

（3）利用 C 语言编译器将目标程序与程序所调用的相关函数和子程序进行链接，最终形成可执行的二进制程序文件。

从上述过程可见，应用程序是从源代码文件开始，经过编译与链接，最终形成可执行的二进制文件的。可见，只要获得应用程序的源代码，就可以通过编译和链接操作获得最终可执行的应用程序了。这就是利用源代码安装应用程序的基本思路。

在了解了应用程序的开发方式之后，为了能够顺利地安装应用程序，首先要解决几个问题。

（1）源代码文件的来源。

（2）如何编译源代码文件。

（3）如何将编译好的目标文件与调用函数或子程序进行链接。

（4）如何将编译好的应用程序安装到操作系统中。

13.1.1　源代码文件的来源

根据 GPL 授权协议的定义，开源软件的提供者在发布应用程序本身的同时，要提供应用程序的源代码文件。这既体现了开源软件的共享精神，又为使用者定制和强化软件的功能提供了很大的便利。

开源软件的开发者所提供的软件源代码文件并不是单一的文本文件，而是一系列文件的集合。它既包括主程序文件，同时也可能包括若干子程序文件与被调用函数等。每个软件的源代码文件的数量和体积是不同的，为了便于在网络中传输，往往将这一系列文件利用 tar 程序打包成一个文件，并利用压缩工具进行压缩（常用的压缩工具是 gzip），这样就形成了打包后压缩的文件，称为 Tarball 文件。

当需要获得某开源软件的源代码文件时，可在该开源软件的主页或 Linux 社区中查找该软件的 Tarball 文件并下载至本地。

13.1.2　如何编译与链接源代码文件

获得软件的源代码文件之后，需要对源代码文件进行编译，并与其所调用的函数和子程序进行链接，这个过程是利用编译器来完成的。由于 Linux 操作系统中的应用程序大多是利用 C 语言进行开发的，所以应用最多的就是 gcc 这款 C 语言编译器。

编译时，需要利用 gcc 命令将 Tarball 中的每个源代码文件进行编译。如果要编译的

应用程序比较小，只有几个源代码文件，那么这种方式还可以接受。但是，如果该应用程序的源代码文件成百上千，则这种编译工作量是非常大的。

编译好每个源代码文件之后，就需要将目标程序与其子程序或相关函数库进行链接了。同样，这个过程也是借助于 gcc 编译程序来完成的。在连接前要明确，该程序有多少链接关系，涉及多少子程序与相关函数。在明确这些概念之后才能一步步地开始链接配置。若程序的体积较大、关系比较复杂，则这种编译工作将变得十分复杂。

为克服这种编译工作对操作者相关开发知识的要求以及简化程序编译工作，Linux 操作系统提供了一个 make 应用程序管理工具，该工具的一个主要功能就是帮助用户编译、链接和安装源代码应用程序。

make 工具首先会查找应用程序源代码中的 Makefile（或 makefile）文件，并按照该文件所定义的源代码文件编译和链接操作规则自动调用 gcc 编译程序完成源代码文件的编译和链接工作，并最终将编译好的应用程序安装到操作系统中。利用该工具，用户只需执行一个 make 命令即可完成源代码文件的编译和链接等一系列复杂的操作，这样大大减少了用户的工作量，使软件源代码的编译和链接工作变得友好、简单。

需要注意的是，make 工具是需要借助于 Makefile（makefile）这个编译规则文件才能完成工作的，而 Makefile 文件是从何而来的呢？

13.1.3　编译规则文件 Makefile

Makefile 文件是一个编译规则文件，该文件中记载了源代码文件编译和链接的规则和方法。make 工具可根据该文件的记载调用 gcc 等相关应用程序完成对软件源代码的编译。

Makefile 文件的产生方式是比较特殊的，它是由 configure（或 config）程序自动建立的。

configure（或 config）程序使用开源软件的开发者提供的一个环境检测工具，该工具一般会随软件源代码文件一并发送。也就是说，当获得软件的 Tarball 文件时，该文件中除包含有软件的源代码文件之外，还包括一个 configure（或 config）工具。

configure 工具的主要作用表现在两个方面。

（1）检测当前操作系统环境，包括硬件平台、操作系统内核版本、操作系统的函数库结构与内容、操作系统的编译器版本等信息，以确定该应用软件可以在当前的操作系统中运行。

（2）生成 Makefile 编译规则文件。configure 工具会根据对操作系统环境的检测情况判断应用程序能否在当前操作系统中顺利编译与运行。如果不能，则直接退出应用。如果能，则 configure 工具会根据操作系统环境生成 Makefile 编译规则文件供 make 程序调用。

13.1.4　软件的安装

软件编译和链接完成后得到的是一个可执行的应用程序，但该应用程序还不能立刻在系统中运行，还需要将相关的资源分配到不同的目录中，以便于程序代码的调用。这

些资源包括应用程序的配置文件、应用程序的可执行文件、相关的函数库以及相关的在线帮助说明等。

这种应用程序资源分配的操作称为软件的安装操作。软件的安装方法，或称为软件资源的分布方法，在 Makefile 文件中已经事先进行了定义，可使用 make 命令直接调用 Makefike 文件的 install 操作来自动执行。

```
[root@localhost local]# make install
```

以上是源代码文件的编译与安装的基本思路，接下来以 Htop（Htop 是一个动态进程监控工具）应用程序的安装为例来说明整个安装过程。

13.2 源代码应用程序安装实例

本例中以 Htop 应用程序的安装为例，说明源代码应用程序的安装方法。

Htop 是一个交互式的进程查看器，以文本模式工作于控制台或 X 终端中。与 top 相比，htop 具有横向或纵向滚动浏览进程列表、快速启动以及支持鼠标操作等优点。Htop 已在很大程度上替代了 top 工具。

Htop 源代码方式安装流程如下。

（1）获得 Htop 的 Tarball 源代码包。

（2）解压 Htop Tarball，获得 Htop 源代码文件。

（3）利用 configure 检测当前操作系统环境，并生成 Makefile 编译规则文件。

（4）利用 make 工具调用 Makefile 文件进行软件的编译与链接。

（5）利用 make 工具调用 Makefile 文件的 install 选项，将编译好的应用程序安装到操作系统中。

13.2.1 获得 Htop 的源代码包 Tarball

Htop 软件的源代码文件可在 Htop 软件的官方站点下载，下载地址为：

```
http://htop.sourceforge.net/index.php?page=downloads#sources
```

当前版本为 htop-0.5.3.tar.gz。

在 Htop 的下载页面可见，Htop 的开发环境为 C/C++，使用的操作系统为 Linux。也就是说 Htop 是一种利用 C 语言开发的可运行于 Linux 操作系统上的应用程序。可以利用 C 语言的编译程序 gcc 对其进行编译与链接。

13.2.2 解压 Htop Tarball

对于利用源代码安装软件而言，软件的源代码按照 FHS 的标准应位于/usr/local/src 目录中。这种存放目录的选择虽然不会对软件的编译与安装产生影响，但是对于有序管理应用软件的作用是很大的。

将 htop-0.5.3.tar.gz 这个 Tarball 移动至“/usr/local/src”目录中，并利用 tart 命令解压

缩与解包。

```
[root@localhost ~]# mv htop-0.5.3.tar.gz /usr/local/src
[root@localhost ~]# cd /usr/local/src
[root@localhost src]# ls
htop-0.5.3.tar.gz
[root@localhost src]# tar -zxvf htop-0.5.3.tar.gz
```

解压后会得到一个名称为 htop-0.5.3 的目录，该目录中包含的软件的源代码文件、configuer 工具程序、README 或 INSTALL 说明文件等内容。

这里要特别注意，README 或 INSTALL 文件是应用程序的安装与使用说明文件。应用程序的安装方法会在该文件中有详细的说明，因此在安装软件前要仔细阅读该文件。

```
[root@localhost src]# cd htop-0.5.3
[root@localhost htop-0.5.3]# ls
 htop.1     htop.c        README
configure    Hashtable.c    Object.h
...
```

13.2.3 执行 configure 程序

在得到源代码文件目录后，进入该目录并执行该目录下的 configure 程序。注意，执行当前目录下的应用程序使用“./可执行文件名”这个命令。

```
[root@localhost htop-0.5.3]# ./configure
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for gawk... gawk
...
```

经过一系列的检测操作后，configure 工具会自动创建 Makefile 这个规则文件。

```
[root@localhost htop-0.5.3]# ls -l Makefile
-rw-r--r-- 1 root root 21474 07-22 09:38 Makefile
```

注意，如果 configure 工具在检测环境时出现错误，则会对用户作出相应的提示。这些错误常见的包括没有编译程序、不能发现函数库以及头文件错误等。

产生这种错误的原因主要是因为在操作系统安装过程中相应的开发组件没有安装。这也提醒了我们，在安装操作系统时，安装开发环境是很重要的。若因为开发环境没有安装而导致检测、编译等操作无法进行的话，也可以通过后面章节所介绍的 rpm 工具或 yum 工具来补全缺失的开发工具。

13.2.4 使用 make 工具开始编译

在获得 Makefile 编译规则文件之后，就可以使用 make 工具开始编译源文件了。

这里需要注意的一点是，源代码文件中可能包含之前已经编译过的目标程序。这些目标程序由于不是在本机编译的，所以其所调用的相关资源也不尽相同。为此，在正式编译之前需要将以前已经编译过的目标程序删除。删除命令如下：

```
[root@localhost htop-0.5.3]# make clean
test -z "htop" || rm -f htop
rm -f *.o
```

上述命令的含义是 make 命令调用 Makefile 文件中的 clean 参数来清除以前编译的目标文件。

在完成清除工作之后，就可以直接利用 make 命令进行编译了。

```
[root@localhost htop-0.5.3]# make
make  all-am
make[1]: Entering directory '/usr/local/src/htop-0.5.3'
…
```

13.2.5 安装软件

编译完成后，就可以将软件安装到操作系统中了。软件的安装使用如下命令：

```
[root@localhost htop-0.5.3]# make install
make[1]: Entering directory '/usr/local/src/htop-0.5.3'
test -z "/usr/local/bin" ‖ mkdir -p -- "/usr/local/bin"
  /usr/bin/install -c 'htop' '/usr/local/bin/htop'
test -z "/usr/local/man/man1" ‖ mkdir -p -- "/usr/local/man/man1"
 /usr/bin/install -c -m 644 './htop.1' '/usr/local/man/man1/htop.1'
make[1]: Leaving directory '/usr/local/src/htop-0.5.3'
```

安装完成后，可以直接使用 htop 命令来启动 htop 应用程序。

```
[root@localhost local]# htop
```

至此就完成了利用软件源代码安装软件的工作。需要注意的是，整个安装流程具有阶梯性，安装的顺序不能发生变化，否则会出现安装无法继续的情况。

还需要注意，上面所介绍的是标准的源代码软件安装方法。对于一些应用软件而言，还可能具有其他安装需求。例如，视频播放软件在安装时会要求用户自行定义支持选项、自定义皮肤等。对于这类软件而言，需要在安装前仔细阅读软件的说明文件。

13.3 RPM 软件包管理

前面使用源码包来进行安装软件，使用这种方式非常方便，原因是它有非常好的跨平台性。除了使用源码包来安装软件之外，还有一种软件包使用得也非常广泛，那就是 RPM 软件包。

13.3.1 什么是 RPM

RPM（Red Hat Package Manager）是 Red Hat 包管理工具。是 Red Hat 公司提出的一种软件包的管理机制。虽然这是 Red Hat 公司提出的思想，但是其设计理念是开放式的，因此被大多数发行版本号所采用，已经成为一种默认的行业标准。

RPM 软件包是一种已经编译好的格式，用户在使用时只需要获取 RPM 软件包，直接使用 rpm 命令进行安装和管理就可以了。

13.3.2 RPM 软件包格式

在 Red Hat Enterprise Linux 中，所有的软件包格式都是 RPM 格式的，可以挂载光盘进行查看，如下所示：

```
[root@xinya Server]# cd /mnt/cdrom/Server/
[root@xinya Server]# ls
a2ps-4.13b-57.2.el5.i386.rpm
acl-2.2.39-6.el5.i386.rpm
acpid-1.0.4-9.el5_4.2.i386.rpm
adaptx-0.9.13-3jpp.1.i386.rpm
adaptx-doc-0.9.13-3jpp.1.i386.rpm
adaptx-javadoc-0.9.13-3jpp.1.i386.rpm
adjtimex-1.20-2.1.i386.rpm
agg-2.4-2.1.i386.rpm
agg-devel-2.4-2.1.i386.rpm
aide-0.13.1-6.el5.i386.rpm
alacarte-0.10.0-1.fc6.noarch.rpm
alchemist-1.0.36-2.el5.i386.rpm
alchemist-devel-1.0.36-2.el5.i386.rpm
alsa-lib-1.0.17-1.el5.i386.rpm
alsa-lib-devel-1.0.17-1.el5.i386.rpm
alsa-utils-1.0.17-1.el5.i386.rpm
…
```

上面列出的软件包都是 RPM 软件包，并且每个软件包的名称都相似。

一个完整的 RPM 软件包通常由 5 部分组成：

（1）软件名称；

（2）版本信息；

（3）编译次数；

（4）系统架构；

（5）后缀名。

下面是以 apache 服务为例查看 httpd 软件包的结果：

```
[root@xinya Server]# ls httpd-2.2.3-43.el5.i386.rpm
httpd-2.2.3-43.el5.i386.rpm
```

```
[root@xinya Server]#
```

在输出结果中，httpd 为软件名称，2.2.3 代表 httpd 这个软件的版本号，43 表示编译次数，el5 表示在 Enterprise Linux 5，i386 表示当前系统的硬件平台，rpm 表示这个软件包的后缀名。

13.3.3 RPM 软件的管理

RPM 软件包在管理时使用 rpm 命令。可以理解为使用这个软件的后缀名来管理这类格式的软件。

rpm 命令的格式如下：

rpm ［选项］ 参数…

1．RPM 软件包的安装与更新

下面以 httpd 软件为例来演示 rpm 命令的使用过程。

可到系统光盘映像中获取软件包，或到 apache 官方网站下载 RPM 格式的软件包。

如果系统中没有安装软件包，可使用 rpm -i 来安装 RPM 软件包。如下所示：

```
[root@xinya opt]# ls
httpd-2.2.3-43.el5.i386.rpm
[root@xinya opt]# rpm -i httpd-2.2.3-43.el5.i386.rpm
[root@xinya opt]#
```

从输出中可以看到，获取到软件包后，使用“rpm -i”命令将 httpd 软件包安装到系统当中。-i 选项的作用是安装指定的软件包。

这种安装方式非常简便，但不易读取安装时的信息。如果指定安装的软件包比较大，在安装过程中，屏幕会出现假死现象，让用户在安装软件时不知道软件安装的具体进度如何。在安装软件时，可以使用“rpm -ivh”命令来解决这个问题。如下所示：

```
[root@xinya opt]# rpm -ivh httpd-2.2.3-43.el5.i386.rpm
warning: httpd-2.2.3-43.el5.i386.rpm: Header V3 DSA signature: NOKEY, key
ID 37017186
Preparing...          ########################################### [100%]
   1:httpd            ########################################### [100%]
[root@xinya opt]#
```

通过比较可以发现，同一个软件包在两次安装时安装显示的效果不同。第二次安装时会出现两个进度条。-v 选项用来在安装软件时显示更加详细的安装信息。-h 选项的作用是用#来显示软件安装的进度。这样可以更直观地看到软件是否成功安装到系统当中。

当系统中已经安装好了所要安装的软件时，会给出提示，如下所示：

```
[root@xinya opt]# rpm -ivh httpd-2.2.3-43.el5.i386.rpm
warning: httpd-2.2.3-43.el5.i386.rpm: Header V3 DSA signature: NOKEY, key
```

```
ID 37017186
Preparing...        ########################################### [100%]
      package httpd-2.2.3-43.el5.i386 is already installed
[root@xinya opt]#
```

在安装软件包的过程中，如果需要同时安装多个软件包，可将需要安装的软件包名依次写到选项之后，软件包名之间用空格隔开。也可以使用通配符来安装，如下所示：

```
#rpm  -ivh  package1.rpm  package2.rpm  package3.rpm …
```

或

```
#rpm  -ivh  *.rpm
```

如果系统中已经安装了指定的软件，但版本已经非常旧了，可以获取新版本的软件包，使用如下命令进行软件包的更新：

rpm -Uvh 软件包名

2. RPM 软件包的查询

在安装软件包之前，可以使用“rpm -q”命令来查询指定的软件包是否已经安装在系统中，如果已经安装完成，可以避免重复安装的过程，或确定指定的软件包是否要更新。还可以在指定软件包安装过程结束后，查询确定指定的软件包是否被正确安装。命令格式如下：

#rpm -q 软件名

-q：仅查询后面指定的软件名称是否已安装。

```
[root@xinya opt]# rpm -q httpd
httpd-2.2.3-43.el5
[root@xinya opt]#
```

需要注意，在查询软件包时，选项-q 后要写软件包名当中的第一部分，而不能写包含版本号等信息的完整的软件包名，否则会导致查询结果为该软件没有安装。如下所示：

```
[root@xinya opt]# rpm -q httpd
httpd-2.2.3-43.el5
[root@xinya opt]# rpm -q httpd-2.2.3-43.el5.i386.rpm
package httpd-2.2.3-43.el5.i386.rpm is not installed
[root@xinya opt]#
```

-q 选项可以组合其他选项进行查询，例如：

-qa：列出所有已经安装在本机 Linux 系统上的软件名称。

```
[root@xinya opt]# rpm -qa
rmt-0.4b41-4.el5
```

```
desktop-backgrounds-basic-2.0-37
man-pages-2.39-15.el5
popt-1.10.2.3-18.el5
libusb-0.1.12-5.1
readline-5.1-3.el5
bzip2-libs-1.0.3-4.el5_2
gdbm-1.8.0-26.2.1
elfutils-libelf-0.137-3.el5
libfontenc-1.0.2-2.2.el5
bzip2-1.0.3-4.el5_2
ORBit2-2.14.3-5.el5
iptables-ipv6-1.3.5-5.3.el5_4.1
vim-common-7.0.109-6.el5
m4-1.4.5-3.el5.1
bc-1.06-21
libexif-0.6.13-4.0.2.el5_1.1
libevent-1.4.13-1
lftp-3.7.11-4.el5
numactl-0.9.8-11.el5
cyrus-sasl-plain-2.1.22-5.el5_4.3
telnet-0.17-39.el5
cdrdao-1.2.1-2
pam_smb-1.1.7-7.2.1
…
```

-qi：列出指定软件的详细信息，包括开发者、版本与说明等。

```
[root@xinya opt]# rpm -qi httpd
Name        : httpd                    Relocations: (not relocatable)
Version     : 2.2.3                    Vendor: Red Hat, Inc.
Release     : 43.el5      Build Date: 2010年03月04日 星期四 22时58分54秒
Install Date: 2011年05月17日 星期二 14时26分17秒   Build Host: x86-001.
build.bos.redhat.com
Group       : System Environment/Daemons   Source RPM: httpd-2.2.3-43.
el5.src.rpm
Size        : 3272321                  License: Apache Software License
Signature   : DSA/SHA1, 2010年03月09日 星期二 17时46分03秒, Key ID
5326810137017186
Packager    : Red Hat, Inc. <http://bugzilla.redhat.com/bugzilla>
URL         : http://httpd.apache.org/
Summary     : Apache HTTP 服务器
Description :
The Apache HTTP Server is a powerful, efficient, and extensible
web server.
[root@xinya opt]#
```

-ql：列出该软件的所有文件与目录的完整路径。

```
[root@xinya opt]# rpm -ql httpd
/etc/httpd
/etc/httpd/conf
/etc/httpd/conf.d
/etc/httpd/conf.d/README
/etc/httpd/conf.d/proxy_ajp.conf
/etc/httpd/conf.d/welcome.conf
/etc/httpd/conf/httpd.conf
/etc/httpd/conf/magic
/etc/httpd/logs
/etc/httpd/modules
/etc/httpd/run
/etc/logrotate.d/httpd
/etc/rc.d/init.d/httpd
/etc/sysconfig/httpd
/usr/bin/ab
/usr/bin/htdbm
/usr/bin/htdigest
/usr/bin/htpasswd
/usr/bin/logresolve
…
```

-qf：后面接文件名，找出指定的文件属于哪一个已安装的软件包。

#rpm -qf 文件名

```
[root@xinya opt]# rpm -qf /etc/httpd/conf/httpd.conf
httpd-2.2.3-43.el5
[root@xinya opt]#
```

3．卸载 RPM 软件包

当某个软件包不再需要或版本太旧，想从系统当中移除该软件包时，可以使用“rpm -e”命令来完成。命令格式如下：

rpm -e 软件名

例如：

```
[root@xinya opt]# rpm -e httpd
[root@xinya opt]# rpm -q httpd
package httpd is not installed
[root@xinya opt]#
```

从输出中可以看出，当卸载了指定的软件包后，使用“rpm -q”命令进行查询，系统提示指定的软件包没有安装。

4. 解决软件包之间的依赖关系

在软件包安装或卸载的过程中，可能会遇到安装或卸载不成功，系统提示在安装或卸载指定软件包前先要安装指定软件包所依赖的软件包。例如：

```
[root@xinya opt]# rpm -e httpd
error: Failed dependencies:
      httpd-mmn = 20051115 is needed by (installed) mod_python-3.2.8-3.1.i386
      httpd-mmn = 20051115 is needed by (installed) mod_ssl-2.2.3-43.el5.i386
      httpd-mmn = 20051115 is needed by (installed) mod_perl-2.0.4-6.el5.i386
      httpd-mmn = 20051115 is needed by (installed) php-5.1.6-27.el5.i386
      webserver is needed by (installed) webalizer-2.01_10-30.1.i386
      httpd >= 2.0.40 is needed by (installed) mod_python-3.2.8-3.1.i386
      httpd = 0:2.2.3-43.el5 is needed by (installed) mod_ssl-2.2.3-43.el5.i386
      httpd = 2.2.3-43.el5 is needed by (installed) httpd-manual-2.2.3-43.el5.i386
[root@xinya opt]#
```

如上所示，在卸载 httpd 软件包时，系统提示 httpd 软件包与上述软件包之间存在依赖关系，如果指定的软件包被删除，则可能会导致一些软件包不能正常使用或出现一些意想不到的错误。

那么在安装或卸载软件包时一旦出现了软件包之间的依赖关系，应该怎么解决呢？

第一种方法比较笨拙，就是按照在操作时的系统提示逐一安装或卸载相应的软件包，解决软件包之间的依赖关系问题，然后再去安装指定的软件包。这种方法非常复杂，因为在安装或卸载指定软件包所依赖的软件包时可能还会出现新的依赖关系，则也要逐一进行解决。显然这种方法比较笨拙，在解决软件包的依赖关系时，可以使用 YUM 工具来自动解决软件包之间的依赖关系问题。

第二种方法是使用“rpm -nodeps”命令，在安装指定软件包时，不考虑软件包之间的依赖关系，直接安装指定的软件包。但这种安装方式可能会造成软件安装完成后无法正常使用的问题。如下所示：

```
[root@xinya opt]# rpm -q httpd
httpd-2.2.3-43.el5
 [root@xinya opt]# rpm -e httpd
error: Failed dependencies:
      httpd-mmn = 20051115 is needed by (installed) mod_python-3.2.8-3.1.i386
      httpd-mmn = 20051115 is needed by (installed) mod_ssl-2.2.3-43.el5.i386
      httpd-mmn = 20051115 is needed by (installed) mod_perl-2.0.4-6.el5.i386
      httpd-mmn = 20051115 is needed by (installed) php-5.1.6-27.el5.i386
      webserver is needed by (installed) webalizer-2.01_10-30.1.i386
      httpd >= 2.0.40 is needed by (installed) mod_python-3.2.8-3.1.i386
      httpd = 0:2.2.3-43.el5 is needed by (installed) mod_ssl-2.2.3-43.el5.i386
      httpd = 2.2.3-43.el5 is needed by (installed) httpd-manual-2.2.3-43.el5.i386
```

```
[root@xinya opt]# rpm -e httpd --nodeps
[root@xinya opt]# rpm -q httpd
package httpd is not installed
[root@xinya opt]#
```

从上例可以看出，第一次在卸载 httpd 软件包时没有卸载成功，系统提示 httpd 软件包与其他软件包存在依赖关系。第二次卸载时使用了--nodeps 选项，忽略了软件包之间的依赖关系，再次查询时可以看到指定的软件包被卸载了。

第14章　Linux 的进程管理

本章的内容是操作系统最重要的部分之一。本章从讲解进程（process）的理论开始，逐步介绍实际操作的管理。本章最重要的目标是提高维护系统的能力。

14.1　进　　程

14.1.1　进程的产生

进程是执行中的程序。而一个程序可以重复执行，产生一个以上的进程。在 Linux 系统中同时有很多进程正在执行，每个进程都有独立的且不重复的编号，叫做进程编号（Process ID，PID），PID 就可以代表特定的进程。Linux 系统的第一个进程为 init，启动第一个执行的进程，PID 为 1。

凡是要执行程序，就是建立一个新进程。建立新进程的方法为 fork（派生），就是从一个已经存在的旧进程分出一个新的进程。分出的新进程为旧进程的子进程（Children Process），而旧进程是父进程（Parent Process）。子进程的各种环境设置都会继承自父进程。

在系统中的每一个进程都是由父进程派生而来，所以都可以往上追溯到第一个进程 init，这是唯一由内核直接执行的进程。而当用户登录后执行的 Shell 就是 init 的子进程，登录后执行的各进程都是由 Shell 进程衍生出来的。

14.1.2　多任务系统

Linux 是一个多任务系统，其意义就是在完成一个工作前可以接受另一个工作，也就是系统可能正在处理多个工作，而用户还会有系统的控制权，可以在提示符后面输入命令。

进程是执行中的程序，但不仅仅是程序，进程会使用系统的资源。操作系统要负责整合分配资源，计算机上的处理器、内存、存储设备及输入/输出设备等资源要分配给所有正在执行的工作，包括不同用户的工作和系统工作等。Linux 会把所有要处理的工作放到一个队列（queue），然后从队列中取出一个工作处理，一段时间后会放回到队列，然后处理下一个工作，之后再处理未完成的工作，一直到工作完成后就从队列中移走，或是工作被终止。其实这样的分配是在不同时间让所有的工作使用资源。在用户输入命令到按下 Enter 键的时间里，系统都一直在处理工作。

系统管理员可以监视系统中所有的用户及所有的进程，并管理进程执行的时间和优先级或将进程终止。

14.1.3 系统执行中的进程

系统中执行的进程有以下 3 种。

（1）Interactive。

由 Shell 开始的进程，即用户所执行的进程，可以在前台或后台执行。通常用户只能管理自己启动的进程，而管理员 root 有权限管理所有进程。

（2）Batch。

安排要执行的进程。由用户或系统安排时间，由系统自动执行，通常是在后台执行。

（3）Daemon。

系统启动时自动启动的进程，通常会一直执行，提供操作系统的一些功能。很多 Daemon 是常驻的服务，如 inetd 等。

14.1.4 显示进程

下面具体介绍系统中的进程。用户可以使用 ps 命令查看自己在 Shell 下执行的进程。

```
[root@xinya ~]# ps
  PID  TTY        TIME        CMD
 3240 pts/2      00:00:00    bash
 6153 pts/2      00:00:00    ps
```

显示内容的各字段如下：

PID：进程编号，每个进程都有自己的编号，且不会重复。可用 PID 来指定进程。

TTY：进程执行时的终端。

TIME：进程已经执行的时间。

CMD：进程的名称，即执行的命令名称。

ps 命令可以显示更多的信息，如果是以 root 的身份使用，可以监听所有系统中的进程，包括由系统执行的进程及其他用户的进程，显示有关进程的完整信息。命令如下：

```
# ps  aux
```

上面的命令显示系统中的所有进程，并显示其完整信息。部分显示内容如下：

```
[root@xinya ~]# ps  aux
USER   PID  %CPU  %MEM  VSZ   RSS   TTY   STAT  START   TIME  COMMAND
root     1   0.0   0.2  2072  556    ?     Ss   14:33   0:01  init [5]
root     2   0.0   0.0     0     0  ?     S<   14:33   0:00  [migration/0]
root     3   0.0   0.0     0     0  ?     SN   14:33   0:00  [ksoftirqd/0]
root     4   0.0   0.0     0     0  ?     S<   14:33   0:00  [watchdog/0]
root     5   0.0   0.0     0     0  ?     S<   14:33   0:00  [events/0]
root     6   0.0   0.0     0     0  ?     S<   14:33   0:00  [khelper]
```

```
root     7  0.0  0.0      0     0 ?     S<   14:33   0:00  [kthread]
root    10  0.0  0.0      0     0 ?     S<   14:33   0:00  [kblockd/0]
root    11  0.0  0.0      0     0 ?     S<   14:33   0:00  [kacpid]
root    72  0.0  0.0      0     0 ?     S<   14:33   0:00  [cqueue/0]
root    75  0.0  0.0      0     0 ?     S<   14:33   0:00  [khubd]
root    77  0.0  0.0      0     0 ?     S<   14:33   0:00  [kseriod]
root   142  0.0  0.0      0     0 ?     S    14:33   0:00  [khungtaskd]
root   143  0.0  0.0      0     0 ?     S    14:33   0:00  [pdflush]
root   144  0.0  0.0      0     0 ?     S    14:33   0:00  [pdflush]
root   145  0.0  0.0      0     0 ?     S<   14:33   0:00  [kswapd0]
root   146  0.0  0.0      0     0 ?     S<   14:33   0:00  [aio/0]
root   315  0.0  0.0      0     0 ?     S<   14:33   0:00  [kpsmoused]
```

各字段说明如下：

USER：执行的用户。

PID：进程编号。

%CPU：CPU 时间与实际时间的比率。

%MEM：内存使用率。

RSS：占用内存大小（KB）。

TTY：进程执行的终端。？表示不占用终端。

STAT：进程的状态，状态代码说明如表 14.1 所示。

表 14.1　进程的状态代码

代码	说　　明
R	执行中的进程；或排在执行的队列中，随时会执行
S	Sleeping（休眠中）
T	追踪或停止的进程
Z	Zombie（僵尸）进程，进程已经暂停或终止（dead），但父进程不知道或是没有妥善消除该进程，就形成了僵尸进程
W	没有固定的 pages 的进程
<	高优先权的进程
N	低优先权的进程

START：开始执行的时间。

TIME：执行进程经过多少时间。

COMMAND：执行的命令名称。

当需要显示特定用户，如 user1 用户的进程时，使用如下命令：

```
ps  -u  user1
```

如果进程很多，可以根据特定条件来排序，以方便观察。例如，以命令名称来排序：

```
ps  --sort  cmd
```

下面对 ps 命令的用法加以说明。

【语法】**ps** [选项]

选项说明如下：

pids：显示指定 PID 的进程，若不只一个，各 PID 间用逗号隔开。

T：列出终端上的所有进程。

A：列出所有的进程。

c：从 task_struct 变量查询命令名称。

e：显示环境变量。

f：以树状格式显示进程与子进程。

h：标题栏不显示进程与子进程。

j：以工作格式显示进程与子进程。

l：长格式显示进程与子进程。

s：显示信号格式。

tty：列出终端 tty 上的进程。

u：显示用户名称及其他信息，包括 CPU 及内存的使用等。

v：vm 格式。

w：宽格式，不打断太长的显示行。

x：显示包括其他终端的进程。

--help：显示 ps 的说明。ps 命令还有很多更复杂的用法，可参阅 Linux 的系统帮助。

--sort：对输出结果进行排序。

14.2 进程的启动与管理

14.2.1 进程的启动与后台执行

用户输入命令，按 Enter 键就启动了进程。每次输入命令至少打开了一个进程。

1. 打开多进程

在 Shell 下执行命令时，Shell 还是一个系统中正在执行的进程，这时候已经打开了多个进程。例如，用 ps 观察进程，会发现列出来至少有 bash 与 ps 两个进程。

当使用管道的时候，实际上就是一次打开了多个进程。例如：

```
# locate  lib|less
```

同时打开 locate 与 less 两个进程，这两个进程的父进程都是 Shell。一些执行的程序也会产生子进程，这样打开的多进程是阶层性的关系。

2. 打开后台进程

执行进程时，如果需要长时间执行，可以将其放到后台执行，这样还能继续在 Shell

下做别的事。后台执行的方法为：

命令语句&

加上&符号，命令就会在后台执行，同时系统会响应后台执行的 PID。例如：

```
[root@xinya ~]# locate lib| less &
[1] 6165
```

3. nohup

在父进程终止时，子进程也会终止，所以在用户注销 Shell 时，所有在 Shell 下执行的进程将全部终止。如果需要在注销后让进程继续执行，就需要使用 nohup 命令。例如：

```
# nohup  ls  -R &
```

在注销后能继续执行命令，忽略 hangup 的信号。注销后，如果执行的命令有输出数据，会输出到 nohup.out。

14.2.2 执行顺序管理

进程在执行时必须要占用系统资源，而所谓的多任务，其实是在很短的时间内将CPU分配给不同的进程使用。如果有某些进程是在处理比较重要的工作，可以指定其优先执行，使较重要的进程能够得到较多（或较久）的使用系统资源的时间，先行完成；不重要的进程也可以设置较低的优先权，让出系统资源。

1. 启动进程时排定优先权

使用 nice 命令来启动进程，可以排定进程的优先权。优先权用数字表示，0 以上的正数表示降低优先权，正值越大优先权越低；负数为提高优先权，负值越大优先权越高。例如，要降低优先权，则执行：

```
# nice  -5  locate  lib > file.list &
```

优先权数字之前的-并不代表负号，而是选项的意思。nice 的内定值为 10，所以再加上 5，优先权为 15。若要提高优先权，则输入：

```
nice  --10  locate lib > file.list&
```

除了 root，一般用户没有权限使用负数，即只能降低优先权，而不能提高优先权。

下面对 nice 命令的用法加以说明。

nice [选项] 命令语句

选项说明如下：

-优先权：设置优先权，范围为-20~19，默认为 10，增加数字（大于 0）为降低优先权，减小数字（小于 0）为提高优先权。

2. 显示优先权

要显示系统中各进程的优先权，输入以下命令：

```
[root@xinya ~]# ps -l
F S   UID   PID  PPID  C PRI  NI  ADDR  SZ     WCHAN   TTY  TIME     CMD
4 S     0  3240  3238  0  76  0-  1478  wait    pts/2      00:00:00  bash
0 T     0  6164  3240  0  77  0-    56  finish  pts/2      00:00:00  locate
0 T     0  6165  3240  0  78  0-  1322  finish  pts/2      00:00:00  less
0 T     0  6170  3240  0  82  5-  1462  finish  pts/2      00:00:00  vi
0 T     0  6172  3240  0  82  5-  1205  finish  pts/2      00:00:00  cat
0 T     0  6173  3240  0  82  5-  1205  finish  pts/2      00:00:00  cat
4 R     0  6174  3240  0  78  0-  1330  -       pts/2      00:00:00  ps
```

其中 NI 字段显示的数字就是代表优先权的 nice 数字。

3. 管理执行中的进程

已经存在的进程也能改变优先权，使用 renice 命令可以修改进程的优先权。该命令中的优先权的表示法与 nice 的设置相同。同样，只有 root 才能提高优先权，一般用户只能降低优先权。renice 可以使用 PID 来指定进程，也可以使用用户名。

（1）指定进程。

例如，提高 3408 进程的优先权，输入：

```
renice  -10  3408
```

与 nice 不同的是，设置优先权的数字前面不需要加-符号，指定的数字就是优先权数字。

（2）指定用户。

renice 也可以指定用户来改变其进程的优先权。例如，降低用户 user1 的进程优先权，输入：

```
renice  5  -u  user1
```

下面对 renice 命令的用法加以说明。

【语法】**renice　优先权　[-p　PID]　[-u　user]　[-g　pgrp]**

选项说明如下：

优先权：直接以正负数指定，与 nice 命令不同的是数字前不加-符号。

-p　PID：指定 PID，-p 可以不加。

-u　user：指定用户。

-p　pgrp：指定组的 GID。

renice 命令用于控制系统中进程的优先权。用户只能降低自己的优先权，root 可以管理所有进程的优先权。

14.2.3 终止进程

如果一个命令的执行时间过长，或者是屏幕上的输出太多，可以使用 Ctrl+C 组合键终止当前正在执行的进程。当需要终止后台进程时需要使用 kill 命令。

1. 终止进程

终止进程只要指定 PID 即可，例如：

```
# kill  1022
```

然后可以用 ps 检查该进程是否还存在：

```
# ps  1022
```

强行终止一个进程后，如果它有子进程，也会终止子进程。有时一些进程无法使用 kill 命令强行终止，因为 kill 命令默认会送出信号 15 给进程，告诉进程终止，但是有些进程会设置为忽略该信号。kill 命令可以送出不同的信号给进程，其中信号 9 为 kill，会强制中断并终止进程。信号 9 是最有力的，一旦执行就无法阻止，例如：

```
# kill  -9  1020
```

2. 终止信号

kill 命令可以列出所有的信号。

```
[root@xinya ~]# kill  -l
 1)  SIGHUP        2)  SIGINT         3)  SIGQUIT        4)  SIGILL
 5)  SIGTRAP       6)  SIGABRT        7)  SIGBUS         8)  SIGFPE
 9)  SIGKILL      10)  SIGUSR1       11)  SIGSEGV       12)  SIGUSR2
13)  SIGPIPE      14)  SIGALRM       15)  SIGTERM       16)  SIGSTKFLT
17)  SIGCHLD      18)  SIGCONT       19)  SIGSTOP       20)  SIGTSTP
21)  SIGTTIN      22)  SIGTTOU       23)  SIGURG        24)  SIGXCPU
25)  SIGXFSZ      26)  SIGVTALRM     27)  SIGPROF       28)  SIGWINCH
29)  SIGIO        30)  SIGPWR        31)  SIGSYS        34)  SIGRTMIN
35)  SIGRTMIN+1   36)  SIGRTMIN+2    37)  SIGRTMIN+3    38)  SIGRTMIN+4
39)  SIGRTMIN+5   40)  SIGRTMIN+6    41)  SIGRTMIN+7    42)  SIGRTMIN+8
43)  SIGRTMIN+9   44)  SIGRTMIN+10   45)  SIGRTMIN+11   46)  SIGRTMIN+12
47)  SIGRTMIN+13  48)  SIGRTMIN+14   49)  SIGRTMIN+15   50)  SIGRTMAX-14
51)  SIGRTMAX-13  52)  SIGRTMAX-12   53)  SIGRTMAX-11   54)  SIGRTMAX-10
55)  SIGRTMAX-9   56)  SIGRTMAX-8    57)  SIGRTMAX-7    58)  SIGRTMAX-6
59)  SIGRTMAX-5   60)  SIGRTMAX-4    61)  SIGRTMAX-3    62)  SIGRTMAX-2
63)  SIGRTMAX-1   64)  SIGRTMAX
```

其中 15 为默认值，kill 命令不指定信号就会送出信号 15。注销 Shell 时系统会将信号 1 传送给所有 Shell 下启动的进程，nohup 启动的命令就会忽略这个信号。信号 9 会无

条件立刻终止进程。

14.2.4 top

top 是一个动态显示和管理系统进程的工具。命令格式如下：

```
top
```

整个屏幕都会显示 top 的信息，每隔几秒就会更新一次。

```
[root@xinya ~]# top
top - 15:21:43 up 48 min,  3 users,  load average: 0.00, 0.03, 0.06
Tasks: 139 total,   2 running, 131 sleeping,   5 stopped,   1 zombie
Cpu(s):  0.0%us,  0.3%sy,  0.0%ni, 99.3%id,  0.3%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:    255412k total,   251208k used,     4204k free,     6800k buffers
Swap:   524280k total,    81312k used,   442968k free,   108080k cached

  ID  USER    PR  NI   VIRT  RES  SHR  S %CPU %MEM   TIME+     COMMAND
6180  root    15   0  2336 1044  804  R  0.3  0.4   0:00.03  top
   1  root    15   0  2072  556  524  S  0.0  0.2   0:01.34  init
   2  root    RT  -5     0    0    0  S  0.0  0.0   0:00.00  migration/0
   3  root    34  19     0    0    0  S  0.0  0.0   0:00.23  ksoftirqd/0
   4  root    RT  -5     0    0    0  S  0.0  0.0   0:00.00  watchdog/0
   5  root    10  -5     0    0    0  S  0.0  0.0   0:00.02  events/0
   6  root    10  -5     0    0    0  S  0.0  0.0   0:00.00  khelper
   7  root    11  -5     0    0    0  S  0.0  0.0   0:00.00  kthread
  10  root    10  -5     0    0    0  S  0.0  0.0   0:00.11  kblockd/0
  11  root    20  -5     0    0    0  S  0.0  0.0   0:00.00  kacpid
  72  root    20  -5     0    0    0  S  0.0  0.0   0:00.00  cqueue/0
  75  root    10  -5     0    0    0  S  0.0  0.0   0:00.02  khubd
  77  root    18  -5     0    0    0  S  0.0  0.0   0:00.01  kseriod
 142  root    22   0     0    0    0  S  0.0  0.0   0:00.00  khungtaskd
 143  root    15   0     0    0    0  S  0.0  0.0   0:00.10  pdflush
 144  root    23   0     0    0    0  S  0.0  0.0   0:00.08  pdflush
 145  root    18  -5     0    0    0  S  0.0  0.0   0:00.38  kswapd0
 146  root    20  -5     0    0    0  S  0.0  0.0   0:00.00  aio/0
 315  root    11  -5     0    0    0  S  0.0  0.0   0:00.00  kpsmoused
 339  root    16  -5     0    0    0  S  0.0  0.0   0:00.00  ata/0
 340  root    16  -5     0    0    0  S  0.0  0.0   0:00.00  ata_aux
 345  root    18  -5     0    0    0  S  0.0  0.0   0:00.00  kstriped
 354  root    20  -5     0    0    0  S  0.0  0.0   0:00.00  ksnapd
 365  root    10  -5     0    0    0  S  0.0  0.0   0:00.65  kjournald
```

top 显示了很多的信息，屏幕分成两个部分：上半部分是系统信息区，显示系统中所有进程的统计信息、CPU 使用状态、内存及 swap 使用状况；下半部分则显示正在执行的各进程详细动态数据。各字段的意义与 ps 命令显示的意义相同。前面没有提到的字段如下：

PRI：进程执行的优先权。

NI：nice 数字，代表进程优先权。

WCHAN：进程的核心功能（Kernel Function）。

SHARE：共享的内存。

top 不仅能显示信息，还能提供管理功能，下面是 top 常用的功能。

1. 退出

执行 top 的时候不能使用 Shell，停止 top 的方式是按 q 键。

2. 说明

按 h 键可以看到 top 的功能键说明：

```
Help for Interactive Commands - procps version 3.2.7
Window 1:Def: Cumulative mode Off.  System: Delay 3.0 secs; Secure mode Off.

  Z,B     Global: 'Z' change color mappings; 'B' disable/enable bold
  l,t,m   Toggle Summaries: 'l' load avg; 't' task/cpu stats; 'm' mem info
  1,I     Toggle SMP view: '1' single/separate states; 'I' Irix/Solaris mode

  f,o     .Fields/Columns: 'f' add or remove; 'o' change display order
  F or O  .Select sort field
  <,>     .Move sort field: '<' next col left; '>' next col right
  R,H     .Toggle: 'R' normal/reverse sort; 'H' show threads
  c,i,S   .Toggle: 'c' cmd name/line; 'i' idle tasks; 'S' cumulative time
  x,y     .Toggle highlights: 'x' sort field; 'y' running tasks
  z,b     .Toggle: 'z' color/mono; 'b' bold/reverse (only if 'x' or 'y')
  u       .Show specific user only
  n or #  .Set maximum tasks displayed

  k,r     Manipulate tasks: 'k' kill; 'r' renice
  d or s  Set update interval
  W       Write configuration file
  q       Quit
          ( commands shown with '.' require a visible task display window )
Press 'h' or '?' for help with Windows,
any other key to continue
```

3. 终止进程

按 k 键，然后输入 PID。显示如下：

```
top - 15:22:46 up 49 min,  3 users,  load average: 0.00, 0.02, 0.06
Tasks: 139 total,   2 running, 131 sleeping,   5 stopped,   1 zombie
Cpu(s):  0.1%us,  0.2%sy,  0.0%ni, 99.7%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
```

```
Mem:    255412k total,  251328k used,    4084k free,    6896k buffers
Swap:   524280k total,   81312k used,  442968k free,  108080k cached
PID to kill: ←此处输入被结束进程的 PID
  PID USER     PR  NI  VIRT  RES  SHR S %CPU %MEM   TIME+   COMMAND
 2559 root     15   0 38612  10m 5744 S  0.1  4.1 0:56.42 Xorg
 2749 root     15   0 77480 4584 3928 S  0.1  1.8 0:01.56 gnome-power-man
 2144 root     18   0  1976  736  656 S  0.0  0.3 0:01.53 hald-addon-stor
 6180 root     15   0  2336 1044  804 R  0.0  0.4 0:00.10 top
    1 root     15   0  2072  556  524 S  0.0  0.2 0:01.34 init
    2 root     RT  -5     0    0    0 S  0.0  0.0 0:00.00 migration/0
    3 root     34  19     0    0    0 S  0.0  0.0 0:00.23 ksoftirqd/0
    4 root     RT  -5     0    0    0 S  0.0  0.0 0:00.00 watchdog/0
    5 root     10  -5     0    0    0 S  0.0  0.0 0:00.02 events/0
    6 root     10  -5     0    0    0 S  0.0  0.0 0:00.00 khelper
    7 root     11  -5     0    0    0 S  0.0  0.0 0:00.00 kthread
   10 root     10  -5     0    0    0 S  0.0  0.0 0:00.11 kblockd/0
```

4. 显示特定用户的进程

如果只显示某个用户的进程，则按 U 键，然后输入用户账户名称。显示如下：

```
top - 15:23:29 up 50 min,  3 users,  load average: 0.05, 0.03, 0.06
Tasks: 139 total,   3 running, 130 sleeping,   5 stopped,   1 zombie
Cpu(s): 50.0%us, 50.0%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:    255412k total,  251388k used,    4024k free,    6960k buffers
Swap:   524280k total,   81312k used,  442968k free,  108076k cached
Which user (blank for all): ←此处输入用户账号
  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+   COMMAND
    1 root      15   0  2072  556  524 S  0.0  0.2    0:01.34 init
    2 root      RT  -5     0    0    0 S  0.0  0.0    0:00.00 migration/0
    3 root      39  19     0    0    0 R  0.0  0.0    0:00.23 ksoftirqd/0
    4 root      RT  -5     0    0    0 S  0.0  0.0    0:00.00 watchdog/0
    5 root      10  -5     0    0    0 S  0.0  0.0    0:00.02 events/0
    6 root      10  -5     0    0    0 S  0.0  0.0    0:00.00 khelper
    7 root      11  -5     0    0    0 S  0.0  0.0    0:00.00 kthread
```

5. 重设优先权

按 r 键重设优先权，然后输入 PID 和优先权数字。显示如下：

```
top - 15:24:12 up 51 min,  3 users,  load average: 0.02, 0.03, 0.06
Tasks: 139 total,   1 running, 132 sleeping,   5 stopped,   1 zombie
Cpu(s):  1.8%us,  3.8%sy,  0.0%ni, 90.3%id,  3.5%wa,  0.3%hi,  0.3%si,  0.0%st
Mem:    255412k total,  251504k used,    3908k free,    7024k buffers
Swap:   524280k total,   81312k used,  442968k free,  108080k cached
PID to renice: ←此处输入 PID 与优先级
```

```
 PID USER    PR  NI  VIRT  RES  SHR  S %CPU %MEM    TIME+   COMMAND
6185 root    15   0  2332  936  704  R  1.9  0.4   0:00.02  top
   1 root    15   0  2072  556  524  S  0.0  0.2   0:01.34  init
   2 root    RT  -5     0    0    0  S  0.0  0.0   0:00.00  migration/0
   3 root    34  19     0    0    0  S  0.0  0.0   0:00.23  ksoftirqd/0
   4 root    RT  -5     0    0    0  S  0.0  0.0   0:00.00  watchdog/0
   5 root    10  -5     0    0    0  S  0.0  0.0   0:00.02  events/0
   6 root    10  -5     0    0    0  S  0.0  0.0   0:00.00  khelper
   7 root    11  -5     0    0    0  S  0.0  0.0   0:00.00  kthread
  10 root    10  -5     0    0    0  S  0.0  0.0   0:00.11  kblockd/0
  11 root    20  -5     0    0    0  S  0.0  0.0   0:00.00  kacpid
  72 root    20  -5     0    0    0  S  0.0  0.0   0:00.00  cqueue/0
  75 root    10  -5     0    0    0  S  0.0  0.0   0:00.02  khubd
  77 root    18  -5     0    0    0  S  0.0  0.0   0:00.01  kseriod
 142 root    15   0     0    0    0  S  0.0  0.0   0:00.00  khungtaskd
 143 root    15   0     0    0    0  S  0.0  0.0   0:00.10  pdflush
 144 root    23   0     0    0    0  S  0.0  0.0   0:00.08  pdflush
 145 root    18  -5     0    0    0  S  0.0  0.0   0:00.38  kswapd0
```

下面对 top 命令的用法加以说明：

【语法】top [选项]

选项说明如下：

-c：命令名称字段，显示完整命令行。

-d dealy：设置画面上数据更新的时间。

-i：不显示暂停与僵尸进程。

-q：随时更新数据。

-s：安全模式，关闭部分 top 命令功能。

-S：显示每一个进程的总 CPU 时间。

top 命令用于动态显示系统状况和执行的进程。数据数秒会更新一次。

14.3 自动执行的工作

执行命令通常是输入命令立刻执行。但是有时也需要在特定时间才执行特定命令，本节介绍在指定时间自动执行特定命令的方法。

14.3.1 设置执行时间

1. 设置执行

使用 at 命令可以设置在指定时间执行特定的命令，命令执行的结果会通过 E-mail 输出到用户的信箱。基本的 at 命令格式为：

```
at  时间
```

时间的格式如表 14.2 所示。

表 14.2　at 命令的时间格式

格　　式	说　　明
hh:mm	例如 17:30 或 1730
hh:mm 月 天 年	月要使用英文单词或简写，年则是 4 位数，如 1730 nov 11 2010
now + 计时	从现在开始计时的时间后，例如 now + 2 hour
midnight	半夜
noon	中午
teatime	午茶时间，下午 4 点

输入要执行命令的方法有以下 3 种。

（1）标准输入。

at 命令会从标准输入读入命令。

```
# at  teatime
```

提示符号会变成 at>，接着输入要执行的命令，并按 Enter 键换行。输入完成后按 Ctrl+D 组合键。

```
[root@xinya ~]# at teatime
at> echo 'It\'s Teatime'
at> date
at> <EOT>
job 5 at 2011-05-15 16:00
```

最后一行会显示排定的工作编号以及执行的时间。

（2）由文件输入。

可以把要执行的命令放在文件内，如文件 job。

```
at  1730  -f  job
```

（3）重定向输入。

刚刚提到，at 是由标准输入读取命令，所以也可以利用重定向的方法。

```
at  1730 < job
```

2. 显示 at 排定的工作

显示已经安排好的工作，输入：

```
at  -l
```

或

```
atq
```

3. 删除排定的工作

不想执行排好的工作时，可以将指定的工作号码删除。例如，要删除工作 5，执行如下命令：

```
at  -d5  Enter
```

或

```
atrm  5
```

4. 系统安排执行时间

如果是没有时效性的工作，可以在系统负担比较小的时候再执行，以节省资源与提高效率。使用 batch 命令安排工作，系统会自动在负载量低的时候执行。输入的方法与 at 相同，可以不指定时间，命令格式如下：

```
batch
```

下面对 at 命令的用法加以说明。

【语法】**at [选项] 时间**

选项说明如下：

-d *n*：删除工作编号 *n*。

-f 文件：由文件读取命令。

-l [*n*]：显示排定的工作，也可以显示指定的工作 *n*。

-m：将完成的工作用 E-mail 传输给用户。

-V：显示版本。

-v：显示执行的时间。

at 命令在指定时间执行命令。命令由标准输入读取，并以 EOF（Ctrl+D 组合键）结束，或是由文件读取。

14.3.2 定期执行

cron daemon 是一个系统中常驻的服务，用于执行例行性的工作，例如，每周一次或每月一次检查磁盘。cron deamon 会每分钟检查一次排定的工作表（crontab），看是否有要执行的命令，所有的输出会通过 E-mail 发送给用户。cron 是维护系统的一个很方便的工具。

1. 使用 cron 的权限

cron 的功能通常开放给一般用户使用，当然除非特别授权，每个用户只能管理自己的 cron。用户也可以限制特定用户使用 cron，限制文件为/etc/cron.allow 和/etc/cron.deny。

如果这两个文件不存在，通常所有用户都能使用 cron。如果 cron.allow 文件存在，就只有列在文件内的用户可以使用 cron；如果 cron.deny 文件存在，列在文件内的用户就不能使用 cron。

2. 管理 crontab

（1）设置 crontab。

固定执行的例行工作安排在 crontab 内。安装或修改 crontab 要使用下面的命令：

```
crontab  -e
```

这个命令会调用 vi 编辑器来编辑执行的清单。下面就是一份清单的内容：

```
0   0   1,15        *   *     fsck  /home
30  6     *     *   07        quota  -a
```

每一行代表一项排定的工作，在指令前面为排定的时间，总共有 5 个字段，以空格作为分隔符，由左到右的含义如表 14.3 所示。

表 14.3 例行工作的执行时间格式

字段	说 明	字段	说 明
分钟	00~59	月	01~12
点钟	0~24	星期	01~07，代表星期一到星期日
号（日）	01~31		

示例的两项工作分别是：

① 每月 1 号和 15 号检查/home 磁盘。

② 每星期日早上 6:30 取得 quota 数据。

也可以先以 crontab 的格式编辑好一个文本文件，例如 mycron，然后输入：

```
crontab  mycron
```

（2）查看 crontab。

查看 crontab，输入：

```
crontab  -l
```

（3）删除 crontab。

删除 crontab，输入：

```
crontab  -r
```

3. 相关文件

每个用户建立的 crontab 会以用户名称保存在/var/spool/cron 目录下。如果要修改 crontab，不能直接编辑该文件，必须要用 crontab 命令来修改。

4. 执行 cron

cron daemon 在启动程序中会自动执行，不能由命令行输入命令来执行。在修改或建

立 crontab 后并不需要重新执行，因为 cron daemon 会每分钟自动检查工作表。

下面对 crontab 命令的用法加以说明。

【语法】**crontab [选项] [文件]**

选项说明如下：

-e：编辑 crontab 文件，会启动 vi 作为编辑工具。

-l：列出用户的 crontab 文件。

-r：删除用户的 crontab 文件。

-u 用户名：root 可以指定管理其他用户的 crontab 文件。

进程管理是 Linux 管理中变化比较大、控制比较细致的部分。本章中就进程的概念、监控与管理进行了讨论。读者要重点掌握 ps 命令与 kill 命令，它们是灵活控制进程的最常用的命令。同时也要注意 at 与 crontab 这两种计划任务的配置方式，这对于后续的自动管理操作有很大的帮助。

第15章 Linux 的启动引导器

GRUB（Grand Unified Boot Loader）是功能强大的启动引导器，不仅可以对各种发行版的 Linux 进行引导，也能够正常引导 PC 上的其他常见的操作系统。

由于 GRUB 的功能强大，已经逐渐取代了过去在 Linux 中使用的 LILO，而成为各 Linux 发行版的默认的启动引导器。

15.1 GRUB 简介

15.1.1 GRUB 与启动引导器

启动引导器是计算机启动过程中运行的第一个软件，通常计算机启动时在通过 BIOS 自检后读取并运行硬盘主引导扇区（MBR）中的启动引导器（Boot Loader）程序，启动引导器再负责加载启动硬盘分区中的操作系统。

如果启动引导器不能正常工作，将导致操作系统不能正常启动，从而造成计算机整体瘫痪，由此可见启动引导器在整个计算机启动过程中的重要性。

通常每个操作系统在安装过程中都要将自带的启动引导器写入硬盘，以便能够进行自身的引导。在 Linux 中常使用 GRUB 作为默认的启动引导器。

15.1.2 GRUB 的功能

GRUB 和其他启动引导器相比有许多独特的功能。

（1）GRUB 提供了真正的命令交互界面，能够使用户以最大的灵活性使用各种参数引导操作系统和收集系统信息。与 LILO 和其他的启动引导器相比，GRUB 提供了更丰富的功能。

（2）GRUB 支持 LBA（Logical Block Addressing ，逻辑块寻址）模式。在使用 LBA 模式之前，许多引导器都遇到了 1024 柱面 BIOS 限制，不能够访问 1024 柱面以后的文件。现在许多新版本的 BIOS 都支持 LBA 模式。只要系统的 BIOS 支持 LBA 模式，GRUB 就能够从 1024 柱面以后的分区中启动操作系统。

（3）GRUB 能够访问 ext2 分区。每次系统启动时 GRUB 访问 ext2 分区中的配置文件/boot/grub/grub.conf。只有当/boot 分区的物理位置发生变化时才需要重新安装 GRUB 到 MBR。

15.2　安装 GRUB

GRUB 作为著名的启动引导器，通常在安装操作系统时会自动安装，如果在系统安装过程中没有选择安装 GRUB，或者是在旧的系统上安装 GRUB 时，可以手工进行安装。

安装 GRUB 有两个层次的含义。

（1）安装 GRUB 软件包：安装 GRUB 软件包仅仅是把 GRUB 所需要使用的文件安装到当前系统，并不能就此使用 GRUB 来引导操作系统启动。

（2）安装 GRUB 到 MBR：只有把 GRUB 安装到硬盘的主引导扇区才能够实现使用 GRUB 来引导系统。

15.2.1　GRUB 软件包的安装

只有在 GRUB 不是当前系统默认的启动引导器时才需要安装 GRUB 软件包。在 CentOS 中 GRUB 的 RPM 安装包在安装光盘的 CentOS 目录中。示例操作系统中的 GRUB 软件版本为 grub-0.97-13.5.i386.rpm。

在获得 RPM 安装包后即可开始整个安装过程了。

首先要查询当前系统是否安装了 GRUB，可以使用如下命令：

```
[root@xinya CentOS]# rpm -q grub
package grub is not installed
```

查询结果显示当前系统没有安装 grub 软件，这时可以使用 rpm 命令安装 grub 软件包。

```
[root@xinya CentOS]# rpm -ivh grub-0.97-13.5.i386.rpm
warning: grub-0.97-13.5.i386.rpm: Header V3 DSA signature: NOKEY, key ID
e8562897
Preparing...          ########################################### [100%]
     package grub-0.97-13.5.i386 is already installed
```

安装完成后再次查询该软件包时，系统提示该软件包已经安装，版本为 0.97-13。

```
[root@xinya CentOS]# rpm -q grub
grub-0.97-13.5
```

15.2.2　安装 GRUB 到 MBR

手工安装 GRUB 到 MBR 需要进行如下两项工作。

（1）建立 GRUB 配置文件。建立文件/boot/grub/grub.conf 并按照主机硬盘中已安装的操作系统进行配置（具体配置方法在稍后将详细介绍）。

（2）使用安装命令安装 GRUB 到 MBR。

命令如下：

```
#grub-install  /dev/硬盘编号
```

如将 GRUB 安装到系统中的第一块 IDE 接口硬盘的 MBR，可以使用以下命令：

```
[root@xinya CentOS]# grub-install /dev/hda
Installation finished. No error reported.
This is the contents of the device map /boot/grub/device.map.
Check if this is correct or not. If any of the lines is incorrect,
fix it and re-run the script 'grub-install'.

# this device map was generated by anaconda
(hd0)     /dev/hda
```

15.3　GRUB 的操作界面

15.3.1　GRUB 的启动菜单界面

正确安装 Linux 操作系统后，可以从硬盘引导系统进入 GRUB 启动菜单界面。首先显示一个默认启动界面，如图 15.1 所示。

```
Press any key to enter the menu

Booting CentOS (2.6.18-194.el5) in 3 seconds...
```

图 15.1　GRUB 启动界面

该界面提示默认将进入“CentOS （2.6.18-194.el5）”这个操作系统。同时，该界面还提示按任意键将进入编辑菜单（menu）。

GRUB 启动菜单界面如图 15.2 所示。

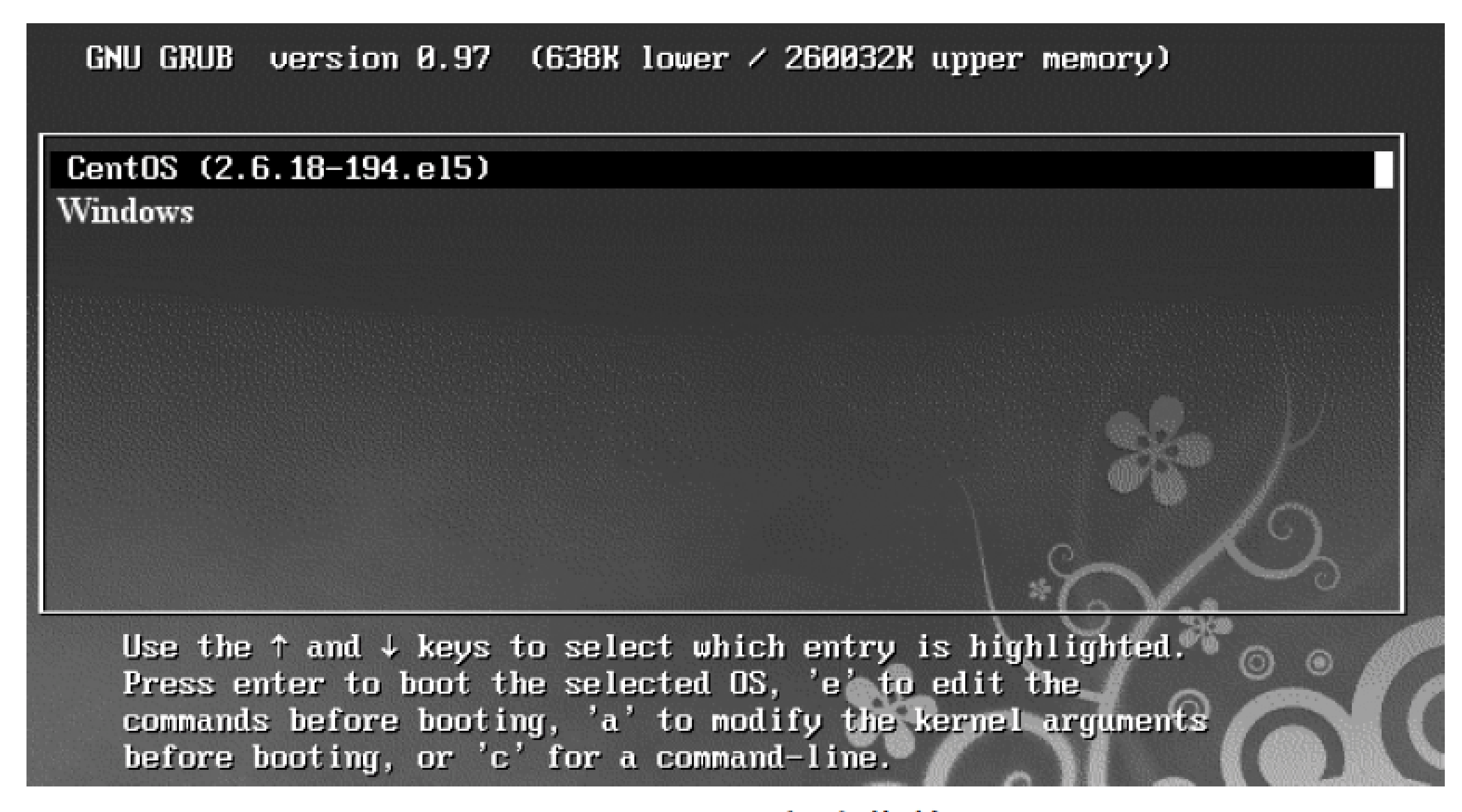


图 15.2　GRUB 启动菜单界面

在该界面中，根据菜单下部的提示，可以使用的编辑键如下：

↑和↓：使用上下箭头键在启动菜单项间进行移动，以便选择启动特定的操作系统。

enter：使用回车键启动当前的菜单项。

e：使用 e 键编辑当前的启动菜单项。

a：使用 a 键添加内核的启动参数。

c：使用 c 键进入 GRUB 的命令方式。

15.3.2 GRUB 的启动菜单项编辑界面

在 GRUB 的启动菜单界面中按 e 键进入 GRUB 的启动菜单项编辑界面，如图 15.3 所示。

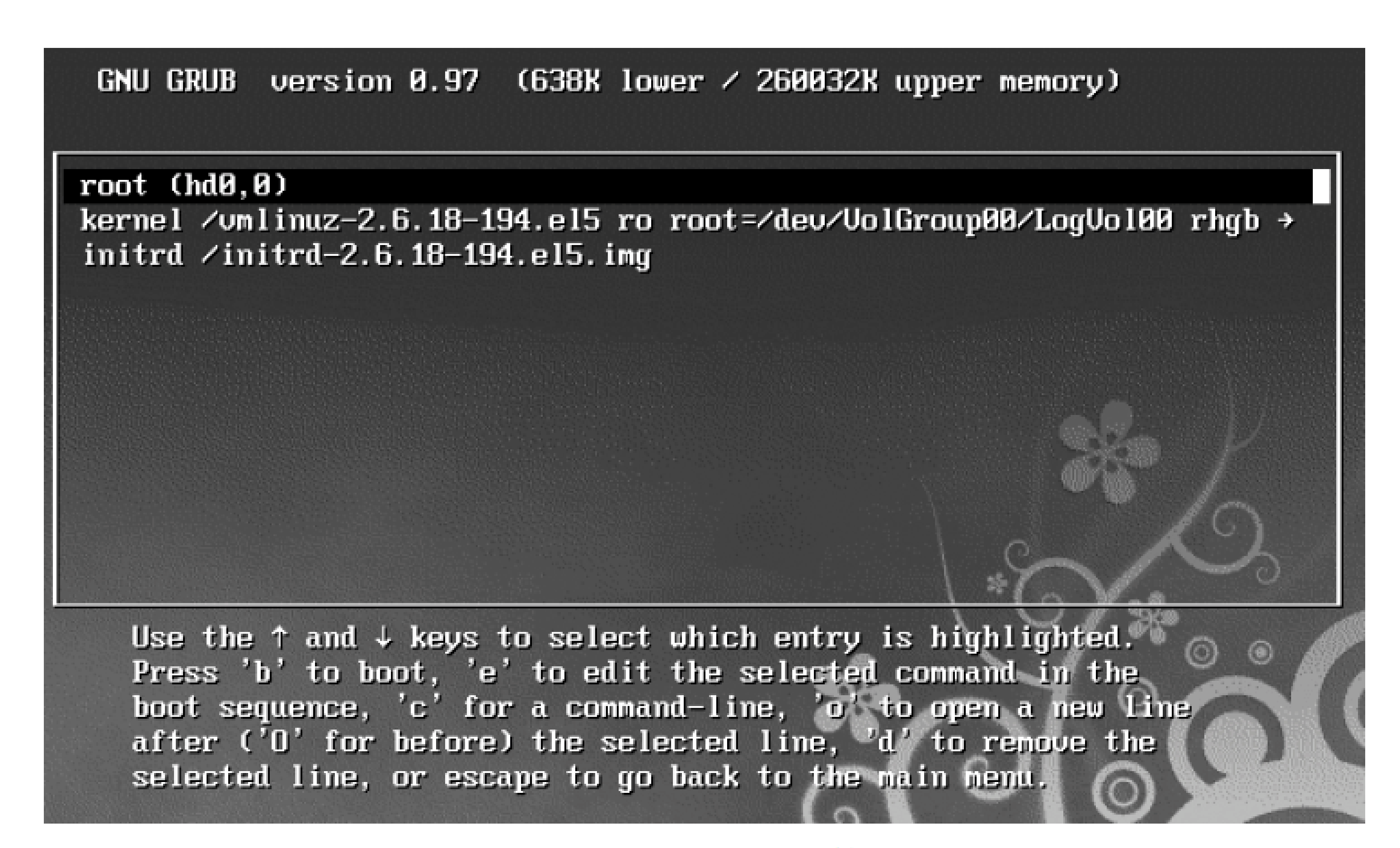


图 15.3 GRUB 的启动菜单项编辑界面

在该界面可以使用的操作按键如下：

↑和↓：使用上下箭头键选择菜单项中的行。

b：使用 b 键启动当前的菜单项。

e：使用 e 键编辑当前选中的行。

c：使用 c 键进入 GRUB 的命令行方式。

o：使用 o 键在当前行后面插入一行。

O：使用 O 键在当前行前面插入一行。

d：使用 d 键删除当前行。

Esc：使用 Esc 键返回 GRUB 启动菜单界面。

在 GRUB 的启动菜单项编辑界面下可以对 GRUB 配置文件中已经存在的启动项做进一步的调整，例如：对现有的命令行进行编辑，添加或删除命令行。最后按 b 键以当前的配置启动。

注意，在 GRUB 的启动菜单项编辑界面下所做的修改只对本次的启动生效，并不保存到配置文件中。如需要改变启动菜单项的配置，可编辑 GRUB 的配置文件 /boot/grub.grub.conf。

15.3.3 GRUB 命令行界面

GRUB 有两种方法可以进入命令行界面：从 GRUB 启动菜单进入命令行界面或者在 Shell 状态下使用 grub 命令进入命令行界面，使用这两种方法获得的命令行界面稍有不

同。由于 grub 命令是运行在 Linux 操作系统中的，由于受到操作系统的限制，使很多命令不能使用；而从 GRUB 启动菜单进入命令行界面支持的命令比较完整。

1. GRUB 命令行界面的特点

GRUB 命令行界面提供了方便友好的命令行交互方式，其主要特点如下。

（1）提供在线帮助命令 help，并且可以获得每条命令的详细帮助。

（2）可以使用左右方向键编辑行命令。

（3）可以使用上下方向键滚动历史命令。

（4）可以使用 Tab 键补全命令和路径。

2. 从 GRUB 启动菜单进入命令行界面

从 GRUB 的启动菜单界面或菜单项编辑界面按 c 键可以进入 GRUB 的命令行界面，如图 15.4 所示。

```
    GNU GRUB  version 0.97  (638K lower / 260032K upper memory)

 [ Minimal BASH-like line editing is supported.  For the first word, TAB
   lists possible command completions.  Anywhere else TAB lists the possible
   completions of a device/filename.  ESC at any time exits.]

grub> 
```

图 15.4　GRUB 的命令行界面

按 Esc 键可以返回菜单界面，使用 help 命令可以获得 GRUB 当前可使用的命令，如图 15.5 所示。

```
grub> help
background RRGGBB                          blocklist FILE
boot                                       cat FILE
chainloader [--force] FILE                 clear
color NORMAL [HIGHLIGHT]                   configfile FILE
displayapm                                 displaymem
find FILENAME                              foreground RRGGBB
geometry DRIVE [CYLINDER HEAD SECTOR [     halt [--no-apm]
help [--all] [PATTERN ...]                 hide PARTITION
initrd FILE [ARG ...]                      kernel [--no-mem-option] [--type=TYPE]
makeactive                                 map TO_DRIVE FROM_DRIVE
md5crypt                                   module FILE [ARG ...]
modulenounzip FILE [ARG ...]               pager [FLAG]
partnew PART TYPE START LEN                parttype PART TYPE
reboot                                     root [DEVICE [HDBIAS]]
rootnoverify [DEVICE [HDBIAS]]             serial [--unit=UNIT] [--port=PORT] [--
setkey [TO_KEY FROM_KEY]                   setup [--prefix=DIR] [--stage2=STAGE2_
splashimage FILE                           terminal [--dumb] [--no-echo] [--no-ed
terminfo [--name=NAME --cursor-address     testvbe MODE
unhide PARTITION                           uppermem KBYTES
vbeprobe [MODE]

grub> 
```

图 15.5　用 help 查询 GRUB 当前可用的命令

如果将某个命令作为 help 命令的参数，可获得该命令的详细帮助说明，如图 15.6 所示。

3. 从 Linux 的 Shell 进入 GRUB 命令行界面

使用 grub 命令也可以进入 GRUB 命令行界面，该命令的完整路径为/sbin/grub。

```
grub> help kernel
kernel: kernel [--no-mem-option] [--type=TYPE] FILE [ARG ...]
    Attempt to load the primary boot image from FILE. The rest of the
    line is passed verbatim as the "kernel command line".  Any
    modules must be reloaded after using this command. The option
    --type is used to suggest what type of kernel to be loaded. TYPE
    must be either of "netbsd", "freebsd", "openbsd", "linux",
    "biglinux" and "multiboot". The option --no-mem-option tells GRUB
    not to pass a Linux's mem option automatically.

grub>
```

图 15.6　用 help 查询命令的详细帮助

```
#grub
```

输入以上命令后，GRUB 命令行界面内容如下：

```
 GNU GRUB  version 0.97  (640K lower / 3072K upper memory)

 [ Minimal BASH-like line editing is supported.  For the first word, TAB
   lists possible command completions.  Anywhere else TAB lists the possible
   completions of a device/filename.]

grub>
```

GRUB 命令行界面提供了类似 bash 的命令行编辑、命令补全和历史命令等功能。当需要结束该命令行界面时，可以使用 quit 命令。

```
grub> quit
```

15.4　GRUB 配置文件

GRUB 的配置文件默认为/boot/grub/grub.conf，在 GRUB 成功安装到硬盘的主引导扇区（MBR）后，只要编辑该文件就可实现对 GRUB 的配置，无需重写 GRUB 到 MBR。

GRUB 配置文件/boot/grub/grub.conf 有两个符号链接文件/boot/grub/menu.lst 和/etc/grub.conf。

用以下命令查看 GRUB 配置文件的内容：

```
[root@xinya ~]# cat /boot/grub/grub.conf
# grub.conf generated by anaconda
#
# Note that you do not have to rerun grub after making changes to this file
# NOTICE:  You have a /boot partition.  This means that
#          all kernel and initrd paths are relative to /boot/, eg.
#          root (hd0,0)
#          kernel /vmlinuz-version ro root=/dev/VolGroup00/LogVol00
#          initrd /initrd-version.img
#boot=/dev/hda
#以#号开头的是注释内容
```

```
default=0
timeout=5
splashimage=(hd0,0)/grub/splash.xpm.gz
hiddenmenu
#以上为全局配置部分

title CentOS (2.6.18-194.el5)
       root (hd0,0)
       kernel /vmlinuz-2.6.18-194.el5 ro root=/dev/VolGroup00/LogVol00 rhgb quiet
       initrd /initrd-2.6.18-194.el5.img
title windows 2008
       rootnoverify(hd0,1)
       chinloader +1
#以上为菜单配置部分
```

15.4.1 GRUB 配置文件的全局命令

GRUB 配置文件中全局配置部分用于定义 GRUB 启动界面的功能。

```
default=0
timeout=5
splashimage=(hd0,0)/grub/splash.xpm.gz
hiddenmenu
```

（1）default：用于定义启动菜单的默认启动项。default=0 表示默认启动菜单配置部分的第一个操作系统。注意，此处从 0 开始记数。如果默认需要启动第二个操作系统，则此处应为 default=1。

default 命令的格式为：

default=默认第一个启动的操作系统
default=第二个启动的操作系统

（2）timeout：用于定义启动默认操作系统的延迟时间，在该时间延迟内用户可以选择要启动的操作系统，若在 timeout 定义的时间内没有进行选择，则启动由 default 命令定义的默认启动的操作系统。

timeout 命令的格式为：

timeout=延迟时间（该时间的单位是秒）

timeout=5 表示延迟时间为 5 秒。

（3）splashimage：用于定义 GRUB 引导程序的背景图片。splashimage=(hd0,0)/boot/grub/splash.xpm.gz 表示当前 GRUB 程序的背景图片为/boot/grub/splash.xpm.gz，该文件必须是 xpm 格式的图片，也可以 gz 压缩文件的形式存在。此处可以放置用户自定义的 GRUB 背景图片，如公司的 LOGO。对于一个图片而言，若不是 xpm 格式，可使用 convert 命令进行图片的转换。

```
convert  源文件 -colors 颜色数 -geometry 分辨率! 输出文件.xpm
```

如将 ladybugs.jpg 文件转换为 xpm 文件的命令如下：

```
#convert  ladybugs.jpg  -colors 14  -geometry 640x480! ladybugs.xpm
```

注意，GRUB 的背景图片的颜色数为 14，分辨率为 640×480。

（4）hiddenmenu：隐藏系统启动菜单，仅显示 timeout 命令定义的计时时间，如图 15.7 所示。

```
Press any key to enter the menu

Booting CentOS (2.6.18-194.el5) in 3 seconds...
```

图 15.7　隐藏系统启动菜单

15.4.2　GRUB 配置文件的菜单项配置命令

下面使用 title 命令开始一个启动菜单项的配置。

```
title CentOS (2.6.18-194.el5)
     root (hd0,0)
     kernel /vmlinuz-2.6.18-194.el5 ro root=/dev/VolGroup00/LogVol00 rhgb quiet
     initrd /initrd-2.6.18-194.el5.img
```

在该配置中，使用了 title、root、kernel 和 initrd 四个命令：

（1）title 命令用于定义启动菜单命令。本例中使用的 title 内容为：

```
title CentOS (2.6.18-194.el5)
```

GRUB 菜单中的显示内容如图 15.8 所示。

```
  GNU GRUB  version 0.97  (638K lower / 260032K upper memory)

 CentOS (2.6.18-194.el5)
```

图 15.8　GRUB 菜单的显示内容

可见，GRUB 启动菜单项显示的内容是由 title 命令定义的。对于 GRUB 而言，当需要引导多个操作系统时，应使用多个 title 来为每个待引导的操作系统定义菜单项。本例中使用了两个 title 命令定义了两个启动菜单，如图 15.9 所示。

```
title CentOS (2.6.18-194.el5)
     root (hd0,0)
     kernel /vmlinuz-2.6.18-194.el5 ro root=/dev/VolGroup00/LogVol00 rhgb quiet
     initrd /initrd-2.6.18-194.el5.img
title windows
     rootnoverify(hd0,1)
     chinloader +1
```

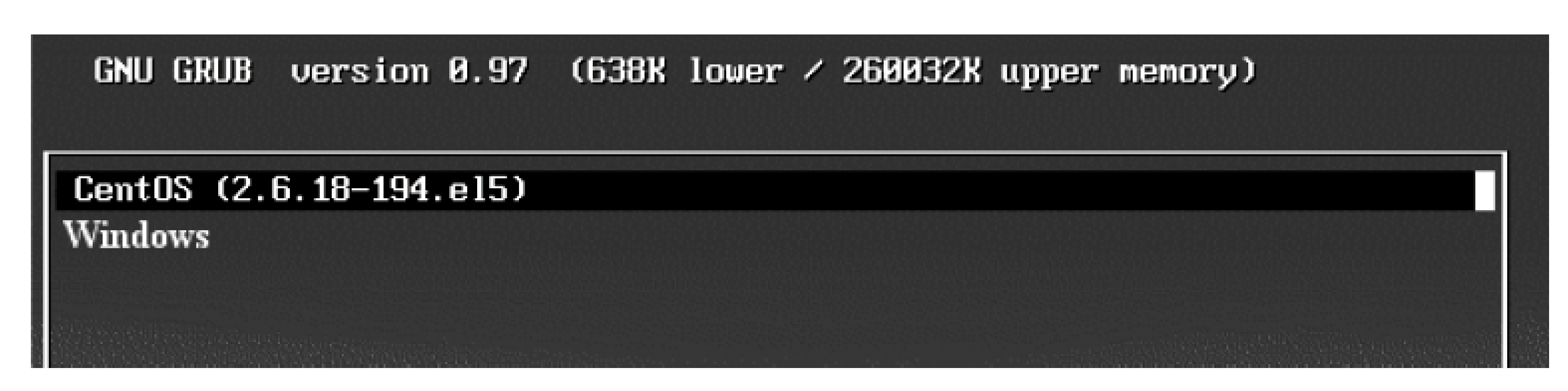


图 15.9　用 title 定义启动菜单项

（2）root 命令用于定义 GRUB 的根设备，即 Linux 内核所在的分区。

这里使用了 root (hd0,0)，表示 Linux 内核在第一块硬盘的第一个分区中。(hd0,0)是硬盘与分区的一种表示方法，在 GRUB 中需要使用这种分区表示方法。这里 hd 是硬盘的意思，“0,0”中的第一个 0 表示的是系统中的第一块硬盘，第二个 0 表示的是硬盘中的第一个分区。注意，此处同 default 命令一样，均是由 0 开始计数。当 Linux 内核在第二块硬盘的第三个分区中时，其表示为(hd1,2)。

（3）kernel 命令用于定义 Linux 的内核文件与加载内核时使用的参数。例如：

```
kernel /boot/vmlinuz-2.6.18-8.el5 ro root=LABEL=/ rhgb quiet
```

表示 Linux 的内核文件为/boot/vmlinuz-2.6.18-8.el5；ro 表示使用只读方式加载内核；root=LABEL=/表示根目录使用卷标“/”表示；rhgb 表示以图形界面方式启动系统，如果不使用 rhgb 则表示使用文字界面启动系统；quiet 命令表示以安静模式启动系统，不显示错误信息。

（4）initrd 命令用于定义初始化内存的镜像文件。

15.4.3　Windows 菜单配置说明

当 GRUB 需要引导其他操作系统时，会使用与引导 Linux 不同的命令。下面以引导 Windows 操作系统为例，来看一下 GRUB 所使用的命令。

```
title windows
      rootnoverify(hd0,1)
      chainloader +1
```

这里使用了 title、rootnoverify 和 chainloader 三个命令。

title：用于定义启动菜单。

rootnoverify：设置 GRUB 的 root 设备，但不加载该文件系统。

chainloader：是间接引导命令，+1 是指将引导程序交由 root 所标示的硬盘的第一个扇区。

以上是 GRUB 对 Windows 的引导配置。

15.5　GRUB 的安全配置

由于 GRUB 负责引导主机中的所有操作系统，作为系统中的第一道屏障，其安全性设置对于整个主机系统是至关重要的。GRUB 提供了较为全面的安全性设置，主要功能

如下。

（1）可针对启动菜单设置全局口令。

（2）可对某菜单项使用全局口令。

（3）可对某菜单项设置独立口令。

（4）对所有的口令都可以使用 MD5 进行加密。

15.5.1 设置全局口令锁定启动菜单

全局口令用于设置只允许用户选择启动菜单项进行启动，如进行其他操作需输入设置的全局口令。

1. password 命令

password 命令用于为 GRUB 的启动菜单和菜单项设置口令。其语法格式为：

【语法】**password 密码**

password 命令可以被应用于 GRUB 配置文件的全局配置部分。在 grub.conf 配置文件的全局配置中使用 password 命令来定义启动菜单口令。

```
default=0
timeout=5
splashimage=(hd0,0)/grub/splash.xpm.gz
hiddenmenu
password xinya  ←定义全局口令
title CentOS (2.6.18-194.el5)
       root (hd0,0)
       kernel /vmlinuz-2.6.18-194.el5 ro root=/dev/VolGroup00/LogVol00 rhgb quiet
       initrd /initrd-2.6.18-194.el5.img
title windows
       rootnoverify (hdo,1)
       chainloader +1
```

使用 password 命令设置了口令之后，重新启动系统后在 GRUB 引导阶段就只能选择要启动的操作系统进行启动了。若需要使用 GRUB 的编辑功能，则需要使用口令解锁启动菜单。

2. 解锁 GRUB 启动菜单

在正确设置了全局口令后，GRUB 启动菜单被锁定，只允许选择菜单项进行启动。如需要对菜单进行其他操作（如编辑菜单项、进入命令行界面等），应先对启动菜单进行解锁，操作步骤如下。

（1）在锁定的启动菜单中按 p 键。

（2）在 Password 提示符后输入正确的口令，并按回车键确认。

（3）如口令输入正确，启动菜单解锁后恢复正常状态。

15.5.2 使用全局口令锁定启动菜单项

GRUB 提供了菜单项级别的保护，对于需要保护的菜单项，可以使用已设置的全局口令进行锁定。如果要启动该菜单项，需要输入全局口令对该菜单项解锁。

锁定启动菜单项的步骤如下。

（1）设置 GRUB 全局口令。

（2）在菜单项配置中使用 lock 命令锁定菜单项。

1. 使用 lock 命令锁定菜单项

lock 命令用于设定某启动菜单项使用全局口令进行锁定，该命令没有参数，一般在 title 命令后使用，锁定该菜单项中 lock 命令之后的所有命令，直到输入正确的口令后，该菜单项才会正常启动。

```
default=0
timeout=5
splashimage=(hd0,0)/grub/splash.xpm.gz
hiddenmenu
password xinya
title CentOS (2.6.18-194.el5)
lock ←锁定该启动菜单
        root (hd0,0)
        kernel /vmlinuz-2.6.18-194.el5 ro root=/dev/VolGroup00/LogVol00 rhgb quiet
        initrd /initrd-2.6.18-194.el5.img
title windows
        rootnoverify (hdo,1)
        chainloader +1
```

注意，用 lock 命令锁定启动项后，在解锁时使用的解锁口令为 password 命令定义的口令。也就是说解锁 GRUB 的编辑功能和解锁启动菜单使用的是相同的口令。

2. 启动被锁定的菜单项

当选择启动被锁定的菜单项时，用户被提示输入口令，如口令验证通过则启动该菜单项的操作系统，如果口令验证失败则屏幕返回启动菜单，如图 15.10 所示。



图 15.10 口令验证失败

15.5.3 设置独立的口令锁定启动菜单

当需要对不同启动菜单项使用不同的口令进行验证管理时，可以在各菜单项中使用

独立的 password 命令设置解锁口令。

```
default=0
timeout=5
splashimage=(hd0,0)/grub/splash.xpm.gz
hiddenmenu
password xinya
title CentOS (2.6.18-194.el5)
password 123456  ←使用 password 命令设置了独立的启动口令
        root (hd0,0)
        kernel /vmlinuz-2.6.18-194.el5 ro root=/dev/VolGroup00/LogVol00 rhgb quiet
        initrd /initrd-2.6.18-194.el5.img
title windows
        rootnoverify (hdo,1)
        chainloader +1
```

这样可实现全局口令和菜单项口令的分级管理，如某菜单项的授权用户可以通过口令验证启动该菜单项，但无权对启动菜单进行其他的操作。

15.6 GRUB 的配置使用技巧

15.6.1 配置 GRUB 重复上次启动项

在切换频繁的多操作系统主机中，在一段时间只启动一种操作系统，而在另一段时间只启动另一种操作系统。为了方便启动，可以把 GRUB 启动菜单的默认启动项设置为该段时期内经常启动的系统。然而经常修改配置文件中的默认启动项也比较麻烦，为此 GRUB 中提供了重复上次启动项的功能，即当前启动菜单的默认启动项为上次启动所选择的菜单项。

具体配置步骤如下。

（1）在全局配置中使用 default saved，用于指定默认启动项为上次保存值。

```
default saved  ←在全局配置中使用"default saved"命令
default=0
timeout=5
splashimage=(hd0,0)/grub/splash.xpm.gz
hiddenmenu
title CentOS (2.6.18-194.el5)
        root (hd0,0)
        kernel /vmlinuz-2.6.18-194.el5 ro root=/dev/VolGroup00/LogVol00 rhgb quiet
        initrd /initrd-2.6.18-194.el5.img
```

（2）在各菜单项配置命令的末尾使用 savedefault 命令，保存当前启动项为默认启动项。

```
default saved
default=0
timeout=5
splashimage=(hd0,0)/grub/splash.xpm.gz
hiddenmenu
title CentOS (2.6.18-194.el5)
        root (hd0,0)
        kernel /vmlinuz-2.6.18-194.el5 ro root=/dev/VolGroup00/LogVol00 rhgb quiet
        initrd /initrd-2.6.18-194.el5.img
        savedefault  ←使用 savedefault 命令定义为默认启动项
```

15.6.2 GRUB 命令参考

GRUB 中的命令可以分为 3 类。

（1）菜单命令：只能用于配置文件的全局配置部分。

（2）常规命令：既能用于配置文件的全局部分，又能在命令行界面使用。

（3）命令行和菜单项命令：既能用于配置文件菜单项定义部分，又能用于命令行界面。

1. 菜单命令

菜单命令只能用于 GRUB 配置文件的全局配置部分，不能用于 GRUB 命令行交互界面。菜单命令在配置文件中应放在其他命令之前。可以使用的菜单命令如下：

default：设置默认启动的菜单项。

fallback：设置启动某菜单项失败后返回的菜单项。

hiddenmenu：隐藏菜单界面。

timeout：设置菜单自动启动的延时时间。

title：开始一个菜单项。

2. 常规命令

常规命令可以应用于配置文件和 GRUB 命令行交互界面。可使用的常规命令如下：

bootp：通过 bootp 初始化网络设备。

color：设置菜单界面的颜色。

device：指定设备文件作为驱动器。

dhcp：通过 DHCP 初始化网络设备。

hide：隐藏某分区。

ifconfig：手工配置网络设备。

pager：改变内部页程序的状态。

partnew：新建一个主分区。

parttype：改变分区的类型。

password：为菜单界面设置口令。

rarp：通过 RARP 初始化网络设备。
serial：设置串口设备。
setkey：设置键盘映射。
splashimage：设置 GRUB 启动的背景图形文件。
terminal：选择终端类型。
tftpserver：指定 TFTP 服务器。
unhide：还原某隐藏分区。

3．命令行和菜单项命令

命令行和菜单项命令可应用于 GRUB 配置文件的菜单项设置中，也可以用在 GRUB 命令行交互界面。常用的命令如下：

blocklist：显示某个文件所在的分区位置。
boot：启动操作系统。
cat：显示文件内容。
chainloader：包启动控制权交给另外的启动引导器。
cmp：比较两个文件。
configfile：加载已存在的 GRUB 配置文件。
debug：设置为 debug 模式。
displayapm：显示 APM BIOS 信息。
displaymem：显示内存配置。
embed：嵌入 Stage 1.5 文件。
find：查找包括某文件的所有设备。
fstest：测试文件系统。
geometry：显示某驱动器的物理信息。
halt：停止计算机的运行（软件关机）。
help：显示 GRUB 的命令帮助信息。
impsprobe：查询对称多处理器（SMP）信息。
initrd：加载 initrd 文件。
install：安装 GRUB。
ioprobe：查询某驱动器的输入输出端口。
kernel：引导操作系统内核。
lock：锁定 GRUB 的引导菜单项，使其输入密码后才可以启动。
makeactive：激活某主分区。
map：映射某虚拟驱动器。
md5crypt：使用 MD5 算法加密口令。
module：加载模块。
modulenounzip：加载模块不进行压缩。
pause：暂停并等待按键。

quit：退出 GRUB。

reboot：重新启动计算机。

read：读取内存中的内容。

root：设置 GRUB 的 root 设备。

rootnoverify：设置 GRUB 的 root 设备单步加载文件系统。

savedefault：保存当前的启动菜单为默认启动项。

setup：自动安装 GRUB。

testvbe：测试 VESA BIOS EXTENSION。

uppermem：强制设置主机上位内存的大小。

vbeprobe：查询 VESA BIOS EXTENSION 信息。

GRUB 管理是系统启动管理的一部分，本章的重点是 GRUB 配置文件的配置与管理。利用 GRUB 的功能，不仅可以引导启动多种操作系统，进而支持多系统环境。更可以按需控制系统启动的安全与个性化选择。

在系统管理工作中常常会遇到由于分区结构的变化导致系统不能正常启动的情况，在掌握了 GRUB 的工作机制后，一旦系统发生与 GRUB 有关的引导故障，就可以利用 Linux 的救援模式或 LiveCD 介质来启动系统，编辑 GRUB 的配置文件，排除系统启动故障。

第 16 章　Linux 的启动与服务

16.1　CentOS 启动过程概述

CentOS 的启动过程与其他 UNIX 操作系统的启动过程基本类似，都经过了以下几个阶段。

（1）主机启动并进行硬件自检后，读取硬盘 MBR 中的启动引导器程序，并进行加载。

（2）启动引导器程序负责引导硬盘中的操作系统，根据用户在启动菜单中选择的启动项的不同，可以引导不同的操作系统启动。对于 Linux 操作系统，启动引导器直接加载 Linux 内核程序。

（3）Linux 的内核程序负责操作系统启动的前期工作，并进一步加载系统的 INIT 进程。

（4）INIT 进程是 Linux 系统中运行的第一个进程，该进程将根据其配置文件执行相应的启动程序，并进入指定的系统运行级别。

（5）在不同的运行级别中，根据系统的设置键启动相应的服务程序。

（6）在启动过程的最后，将运行控制台程序，提示并允许用户输入账号和口令进行登录。

CentOS 中默认安装的是 GRUB 启动引导器，在本书的前面章节中已经做了较详细的讨论，在此只对 GRUB 配置文件中对 Linux 内核的引导设置进行简单说明。

在 grub.conf 文件中，Linux 内核的引导是通过 kernel 命令完成的。

```
title CentOS (2.6.18-194.el5)
        root (hd0,0)
        kernel /vmlinuz-2.6.18-194.el5 ro root=/dev/VolGroup00/LogVol00 rhgb quiet
        initrd /initrd-2.6.18-194.el5.img
```

kernel 定义的 Linux 内核引导文件在/boot 目录中，在定位内核文件之后，内核将启动 init 进程完成系统启动过程。

```
[root@xinya ~]# ls -l /boot
-rw-r--r--   1  root  root    69593  2010-04-03   config-2.6.18-194.el5
drwxr-xr-x   2  root  root     1024  05-15 16:46  grub
-rw-------   1  root  root  3275197  03-14 22:33  initrd-2.6.18-194.el5.img
```

```
drwx------   2  root  root    12288  03-14 22:19   lost+found
-rw-r--r--   1  root  root    80032  2009-03-13    message
-rw-r--r--   1  root  root   110979  2010-04-03    symvers-2.6.18-194.el5.gz
-rw-r--r--   1  root  root   967675  2010-04-03    System.map-2.6.18-194.el5
-rw-r--r--   1  root  root        0  05-14 21:33   testr
-rw-r--r--   1  root  root  1875796  2010-04-03    vmlinuz-2.6.18-194.el5
```

16.2　INIT 进程

INIT 进程是由 Linux 内核引导运行的，是系统中运行的第一个进程，其进程号 PID 永远为 1。INIT 进程运行后将按照其配置文件引导运行系统所需的其他进程，INIT 进程将作为这些进程的父进程。

```
[root@xinya ~]# ps -ef | head
UID        PID  PPID C  STIME  TTY         TIME       CMD
root         1    0  0  16:03  ?         00:00:00  init [5]
root         2    1  0  16:03  ?         00:00:00  [migration/0]
root         3    1  0  16:03  ?         00:00:00  [ksoftirqd/0]
root         4    1  0  16:03  ?         00:00:00  [watchdog/0]
root         5    1  0  16:03  ?         00:00:00  [events/0]
root         6    1  0  16:03  ?         00:00:00  [khelper]
root         7    1  0  16:03  ?         00:00:00  [kthread]
root        10    7  0  16:03  ?         00:00:00  [kblockd/0]
root        11    7  0  16:03  ?         00:00:00  [kacpid]
```

16.2.1　INIT 的配置文件/etc/inittab

INIT 配置文件的全路径名为/etc/inittab，INIT 进程运行后将按照该文件的配置内容运行系统启动程序。

```
[root@xinya ~]# cat /etc/inittab | grep -v ^#
id:5:initdefault:
si::sysinit:/etc/rc.d/rc.sysinit
l0:0:wait:/etc/rc.d/rc 0
l1:1:wait:/etc/rc.d/rc 1
l2:2:wait:/etc/rc.d/rc 2
l3:3:wait:/etc/rc.d/rc 3
l4:4:wait:/etc/rc.d/rc 4
l5:5:wait:/etc/rc.d/rc 5
l6:6:wait:/etc/rc.d/rc 6
ca::ctrlaltdel:/sbin/shutdown -t3 -r now
pf::powerfail:/sbin/shutdown -f -h +2 "Power Failure; System Shutting Down"
pr:12345:powerokwait:/sbin/shutdown -c "Power Restored; Shutdown Cancelled"
1:2345:respawn:/sbin/mingetty tty1
```

```
2:2345:respawn:/sbin/mingetty tty2
3:2345:respawn:/sbin/mingetty tty3
4:2345:respawn:/sbin/mingetty tty4
5:2345:respawn:/sbin/mingetty tty5
6:2345:respawn:/sbin/mingetty tty6

x:5:respawn:/etc/X11/prefdm -nodaemon
```

16.2.2 inittab 文件解析

inittab 文件作为 INIT 进程的配置文件，用于描述系统启动时和正常运行中将运行哪些进程，在该文件中除注释行（以#号开头的行）外，每一行都具有下面的格式。

```
Id:runlevels:action:process
```

inittab 文件中的每一行是一个设置记录，每个记录中有 id、runlevels、action 和 process 四个字段，每个字段间使用“:”分隔，它们共同确定了某进程在哪些运行级别以何种方式运行。

1. id

id 字段用于在 inittab 文件中唯一标识一个配置记录，可以由 1~4 个字符组成，可以把 id 理解成一个配置记录的名字。

```
x:5:respawn:/etc/X11/prefdm -nodaemon
```

该记录的 id 为 x，该记录为 X 登录设置。

2. runlevels

runlevels 字段用于指定该记录在哪些运行级别中运行。runlevels 可以是单个运行级别，也可以是运行级别列表。

有关系统的运行级别的内容将在稍后做专门的讨论。

```
1:2345:respawn:/sbin/mingetty tty1
```

该项配置在系统的 2、3、4、5 四个运行级别中都运行。

3. action

action 字段描述记录将执行哪种类型的动作。下面对 action 字段的常见设置进行介绍。

（1）initdefault 用于标识系统启动后将进入哪个运行级别，process 字段将被忽略。inittab 文件中如不存在 initdefault 记录，init 进程将会在控制台询问要进入的运行级别。

```
id:5:initdefault:
```

系统默认进入运行级别 5。

（2）sysinit 类进程将在系统启动时在任何 boot 或 bootwait 类进程之前运行，记录中的 runlevels 字段将被忽略。

```
si::sysinit:/etc/rc.d/rc.sysinit
```

系统启动时将使用 rc.sysinit 进行系统初始化。

（3）wait 类进程将在进入指定运行级别后运行一次，init 进程将等待其结束。

```
l0:0:wait:/etc/rc.d/rc 0
l1:1:wait:/etc/rc.d/rc 1
l2:2:wait:/etc/rc.d/rc 2
l3:3:wait:/etc/rc.d/rc 3
l4:4:wait:/etc/rc.d/rc 4
l5:5:wait:/etc/rc.d/rc 5
l6:6:wait:/etc/rc.d/rc 6
```

在进入系统 0~6 运行级别后将执行相应的命令。

（4）ctrlaltdel 用于指定用户使用 Ctrl+Alt+Del 组合键时系统所进行的操作，如重新启动、进入单用户模式等。

```
ca::ctrlaltdel:/sbin/shutdown -t3 -r now
```

设置 Ctrl+Alt+Del 组合键重启有效。

（5）powerfail 用于指定当 UPS 发来断电信号时所运行的命令，powerokwait 用于指定当供电恢复时所运行的命令。

（6）respawn 类进程在结束后会重新启动运行（如控制台登录程序 getty）。

```
1:2345:respawn:/sbin/mingetty tty1
2:2345:respawn:/sbin/mingetty tty2
3:2345:respawn:/sbin/mingetty tty3
4:2345:respawn:/sbin/mingetty tty4
5:2345:respawn:/sbin/mingetty tty5
6:2345:respawn:/sbin/mingetty tty6
```

在 2、3、4、5 运行级别中都会启动 6 个虚拟控制台，当用户退出登录后 mingetty 将重新运行。

4. process

process 字段所设置的是启动进程所执行的命令。

```
si::sysinit:/etc/rc.d/rc.sysinit
```

系统初始化所执行的是/etc/rc.d/rc.sysinit 脚本。

16.2.3　系统运行级别

在 UNIX 系统中通常有 0~6 共 7 个运行级别，各运行级别的含义如表 16.1 所示。

表 16.1　系统的运行级别

系统运行级别	说　　明
0	停机，不要把系统默认运行级别设置为 0，否则系统将不能正常启动
1	单用户模式，用于 root 用户进行系统维护，不允许其他用户使用主机
2	多用户模式，在该模式下不能使用 NFS
3	完全多用户模式，主机作为服务器时通常运行在该模式下
4	未分配使用
5	图形登录的多用户模式，用户在该模式下可进行图形登录
6	重新启动，不要把系统的默认级别设置为 6，否则系统将不能正常启动

1. 查看系统运行级别

查看系统运行级别可以使用 runlevel 命令，该命令用于显示系统当前和上一次的运行级别；如系统中不存在上一次的运行级别，则用 N 代替。

```
[root@xinya ~]# runlevel
N 5
```

系统当前的运行级别为 5，没有上一次运行级别（用 N 表示）。

2. 使用 init 命令转换运行级别

当需要对当前的运行级别进行转换时，可以使用 init 命令。init 命令后应添加运行级别作为参数。

```
init  0123456
```

如将当前的运行级别转换为 2，命令如下。

```
[root@xinya ~]# runlevel
N 5
[root@xinya ~]# init 2
```

执行“init 2”命令后会在系统控制台中显示相应的停止启动服务信息。

1）init 0

init 0 命令用于关机，因为运行级别 0 为停机状态，所以从任何运行级别转换为运行级别 0 都是进行关机操作。

2）　init 6

init 6 命令用于对系统进行重新启动，因为运行级别 6 代表重新启动，所以从任何运行级别转换为运行级别 6 都是进行重新启动操作。

16.2.4 系统初始化脚本

在 Linux 中，inittab 文件中指定使用 rc.sysinit 作为系统的初始化脚本。

```
[root@xinya ~]# grep sysinit /etc/inittab
si::sysinit:/etc/rc.d/rc.sysinit
```

该脚本用于初始化系统配置。该脚本的内容较长，功能也很复杂，有兴趣的用户可以自行阅读。

16.3 Linux 的独立服务程序

Linux 中的服务程序有两种：独立运行的服务程序和受 xinetd 管理的服务程序。更有趣的是 xinetd 也是作为系统中一个独立的服务而运行的。本节主要讨论系统中独立运行的服务。

16.3.1 服务器的启动脚本

Linux 中的每个服务都会有相应的服务器启动脚本，该脚本不仅负责启动服务，还可以用于停止服务、重新启动服务和查询服务状态等任务。

所有的服务器启动脚本都放在目录/etc/rd.d/init.d/中，脚本名称与服务器名称相对应。服务器脚本中大多有简要的功能说明和使用方法。

```
[root@xinya ~]# ls /etc/init.d
acpid                haldaemon        multipathd       restorecond
anacron              halt             netconsole       rpcgssd
apmd                 hidd             netfs            rpcidmapd
atd                  hplip            netplugd         rpcsvcgssd
auditd               iptables         network          sshd
autofs               irda             NetworkManager   syslog
avahi-daemon         irqbalance       nfs              tcsd
avahi-dnsconfd       isdn             nfslock          vncserver
bluetooth            killall          nscd             wdaemon
cups-config-daemon   krb524           portmap          winbind
dnsmasq              kudzu            psacct           wpa_supplicant
dund                 lisa             rawdevices       xfs
firstboot            lm_sensors       rdisc            xinetd
functions            lvm2-monitor     readahead_early  ypbind
gpm                  microcode_ctl    readahead_later
```

该目录中存在哪些脚本与当前系统中所安装的服务器程序有关。同时，每个脚本均包含有简要的功能说明和使用方法。

```
[root@xinya ~]# head /etc/rc.d/init.d/xinetd
```

```
#!/bin/bash
#
# xinetd       This starts and stops xinetd.
#
# chkconfig: 345 56 50
# description: xinetd is a powerful replacement for inetd. \
#             xinetd has access control mechanisms, extensive \
#             logging capabilities, the ability to make services \
#             available based on time, and can place \
#             limits on the number of servers that can be started, \
```

16.3.2 各运行级别的脚本目录

系统的各运行级别有独立的脚本目录，目录名称格式为 rc*n*.d，其中 *n* 为 0~6 的数字，对应着各自的运行级别。

```
[root@xinya ~]# ls -l /etc/rc.d
总计 112
drwxr-xr-x  2  root  root   4096  2012-01-21    init.d
-rwxr-xr-x  1  root  root   2255  2009-07-04    rc
drwxr-xr-x  2  root  root   4096  03-17 07:31   rc0.d
drwxr-xr-x  2  root  root   4096  2012-01-21    rc1.d
drwxr-xr-x  2  root  root   4096  2012-01-21    rc2.d
drwxr-xr-x  2  root  root   4096  2012-01-21    rc3.d
drwxr-xr-x  2  root  root   4096  2012-01-21    rc4.d
drwxr-xr-x  2  root  root   4096  2012-01-21    rc5.d
drwxr-xr-x  2  root  root   4096  03-17 07:31   rc6.d
-rwxr-xr-x  1  root  root    220  2009-07-04    rc.local
-rwxr-xr-x  1  root  root  27476  2009-09-29    rc.sysinit
```

16.3.3 服务程序的启动与停止

服务程序不仅可以在系统启动或进入某运行级别时启动或停止，在系统运行过程中用户也可以使用相应的命令直接对某服务进行操作。

1. 各运行级别目录中的脚本

各运行级别的脚本目录中都存在着相应服务程序的脚本，目录中的脚本可分为两类：用于启动服务的和用于停止服务的。

```
[root@xinya ~]# ls /etc/rc.d/rc3.d
```

2. 用于启动服务的脚本

用于启动服务的脚本名格式为“S*nn* 服务名称”，S 是 Start 的缩写，代表启动服务；

nn 为两位数的数字序号，用于确定同类脚本的执行顺序。当系统进入某运行级别时，将按照序号从小到大的顺序执行脚本启动服务。

如某服务在某运行级别中需要运行，则该服务在该运行级别的目录中存在 S 开头的脚本。运行级别目录中的脚本其实都是指向服务器脚本目录/etc/rc.d/init.d/中的符号链接。

```
[root@xinya ~]# ll /etc/rc.d/rc3.d/S56xinetd
lrwxrwxrwx 1 root root 16 03-17 07:31 /etc/rc.d/rc3.d/S56xinetd -> ../init.
d/xinetd
```

3. 用于停止服务的脚本

用于停止服务的脚本名格式为“K*nn* 服务名称”，K 是 Kill 的缩写，代表停止服务；*nn* 为两位的数字序号，用于确定同类脚本的执行顺序。当系统进入某运行级别时，将按照序号从小到大的顺序执行脚本停止服务。

如某服务在某运行级别中不需要运行，则该服务在该运行级别的目录中存在 K 开头的脚本。运行级别目录中的脚本其实都是指向服务器脚本目录/etc/rc.d/init.d/中的符号链接。

```
[root@xinya ~]# ll /etc/rc.d/rc3.d/K89pand
lrwxrwxrwx 1 root root 14 03-14 22:33 /etc/rc.d/rc3.d/K89pand -> ../init.
d/pand
```

4. rc0.d 中的脚本

rc0.d 目录中的脚本用于系统关机，在该目录中的所有脚本都用于停止系统服务，S00Killall 脚本用于停止所有系统进程，S01halt 脚本用于关机。

用户不要人为地去改变 rc0.d 目录中的脚本，否则可能造成系统关机不正常。

5. rc6.d 中的脚本

rc6.d 目录中的脚本用于系统重新启动，在该目录中所有脚本都用于停止系统服务，S00killall 脚本用于停止所有系统进程，S01reboot 脚本用于重新启动系统。

用户不要人为地去改变 rc6.d 目录中的脚本，否则可能造成系统重新启动不正常。

6. 使用服务脚本

在命令行中输入服务脚本的全路径名，后跟相应的动作，可以对服务进行相应的操作。

以 xinetd 服务为例，直接执行服务脚本将显示相应的帮助信息。

```
[root@xinya ~]# /etc/rc.d/init.d/xinetd
用法:/etc/rc.d/init.d/xinetd {start|stop|status|restart|condrestart|reload}
```

提示中显示了服务的控制参数，包括 start（启动）、stop（停止）、restart（重新启动）、

reload（重新加载）和 status（状态）等。

当需要查询服务状态时，可以使用 status 参数。

```
[root@xinya ~]# /etc/rc.d/init.d/xinetd status
xinetd (pid  2255) 正在运行...
```

当然，也可以使用 start、stop 和 restart 参数来启动、关闭和重启服务。

7. 使用 service 命令

在对服务进行操作时每次都输入脚本的全路径名确实比较麻烦，为此有些 Linux 发行版中专门提供了 service 命令解决这个问题。

service 命令用于对系统中的服务进行操作，service 将自动到/etc/rc.d/init.d 目录中查找并执行相应的服务脚本。

service　服务名称　服务动作

以 xinetd 服务为例，可以使用 service 命令来查看该服务的状态、启动或停止服务。

```
[root@xinya ~]# service xinetd status
xinetd (pid  2255) 正在运行…
[root@xinya ~]# service xinetd stop
停止 xinetd:                                              [确定]
[root@xinya ~]# service xinetd start
启动 xinetd:                                              [确定]
```

16.4　xinetd 与其管理的服务

xinetd 作为 inetd 的后续版本，负责管理系统中不频繁使用的服务，这些服务程序在有请求时才由 xinetd 服务负责启动运行，一旦完成服务请求，服务程序即结束运行，这样可有效地减少对系统资源的占用率。

16.4.1　xinetd 的配置文件

xinetd 的配置文件为/etc/xinetd.conf，在该文件中对 xinetd 的默认参数进行配置。

用以下命令查看 xinetd.conf 的内容：

```
[root@xinya ~]# more /etc/xinetd.conf
#
# This is the master xinetd configuration file. Settings in the
# default section will be inherited by all service configurations
# unless explicitly overridden in the service configuration. See
# xinetd.conf in the man pages for a more detailed explanation of
# these attributes.
```

```
defaults
{
# The next two items are intended to be a quick access place to
# temporarily enable or disable services.
#
#       enabled         =
#       disabled        =

# Define general logging characteristics.
        log_type        = SYSLOG daemon info
        log_on_failure  = HOST
        log_on_success  = PID HOST DURATION EXIT
…
```

16.4.2 xinetd 的启动配置目录

xinetd 的启动配置目录为/etc/xinetd.d/，在该目录中 xinetd 管理的每个服务都有独立的配置文件，配置文件的名称与服务名称相同，这是与 inetd 不同的。

在相应的配置文件中对 xinetd 服务将如何启动该服务进行了设置。

```
[root@xinya ~]# ls /etc/xinetd.d/
chargen-dgram    discard-stream   gssftp         rsync
chargen-stream   echo-dgram       klogin         tcpmux-server
daytime-dgram    echo-stream      krb5-telnet    time-dgram
daytime-stream   eklogin          kshell         time-stream
discard-dgram    ekrb5-telnet     ktalk
```

用以下命令查看 rsync 服务的启动配置文件。

```
[root@xinya ~]# more /etc/xinetd.d/rsync
# default: off
# description: The rsync server is a good addition to an ftp server, as it \
#       allows crc checksumming etc.
service rsync
{
        disable         = yes
        socket_type     = stream
        wait            = no
        user            = root
        server          = /usr/bin/rsync
        server_args     = --daemon
        log_on_failure  += USERID
}
```

16.5 服务的启动状态配置命令

在对 Linux 主机的管理中，经常需要设置某服务在某运行级别中自动启动或不启动。在 Linux 中系统提供了 chkconfig 命令可以完成对服务启动的设定工作。

chkconfig 命令的功能强大，可以设置系统中所有服务在运行级别中的启动状态，当然也包括受 xinetd 服务管理的服务。

1. 查看服务启动状态

查看服务启动状态可以使用 chkconfig 命令。

chkconfig --list 【服务名】

chkconfig --list 如果不指定任何参数，则显示所有服务的启动状态；如指定服务名称，则只显示该服务的启动状态。

用以下命令查看系统中所有服务的启动状态。

```
[root@xinya ~]# chkconfig --list
NetworkManager  0:关闭  1:关闭  2:关闭  3:关闭  4:关闭  5:关闭  6:关闭
acpid           0:关闭  1:关闭  2:启用  3:启用  4:启用  5:启用  6:关闭
anacron         0:关闭  1:关闭  2:启用  3:启用  4:启用  5:启用  6:关闭
apmd            0:关闭  1:关闭  2:启用  3:启用  4:启用  5:启用  6:关闭
atd             0:关闭  1:关闭  2:启用  3:启用  4:启用  5:启用  6:关闭
auditd          0:关闭  1:关闭  2:启用  3:启用  4:启用  5:启用  6:关闭
```

用以下命令查看独立服务的启动状态。

```
[root@xinya ~]# chkconfig --list rsync
rsync          关闭
```

2. 设置独立服务的启动状态

当需要设置独立服务在不同运行级别中的启动状态时，可以使用 chkconfig 命令来设置。

chkconfig --level <运行级别列表> <服务名称> <on|off|reset>

设置服务 syslog 在运行级别 2 和 4 中的启动状态为 off，即不启动。

```
[root@xinya ~]# chkconfig --list syslog
syslog         0:关闭  1:关闭  2:启用  3:启用  4:启用  5:启用  6:关闭
[root@xinya ~]# chkconfig --level 24 syslog off
[root@xinya ~]# chkconfig --list syslog
syslog         0:关闭  1:关闭  2:关闭  3:启用  4:关闭  5:启用  6:关闭
```

3. 设置非独立服务的启动状态

由于非独立服务依赖于 xinetd 服务进行启动，所以不存在运行级别启动状态的问题。在使用 chkconfig 命令设置启动状态时也无须指定其运行级别。

chkconfig 服务名称 <on|off|reset>

以 rsync 服务为例：

```
[root@xinya ~]# chkconfig --list rsync
rsync           关闭
[root@xinya ~]# chkconfig rsync on
[root@xinya ~]# chkconfig --list rsync
rsync           启用
```

Linux 操作系统的启动过程会涉及很多的脚本与服务，其主要目的是为用户构建一个可以直接运行程序的环境。

启动过程涉及运行级别、系统初始化与服务控制等几个方面的因素，同时利用 chkconfig 也可以自定义在各运行级别中需要启动的服务。本章的一个难点是 xinetd 所管理的服务的控制，读者应结合 xinetd 配置文件对这种服务的启动、停止以及加载有一个明确的认识。

第17章　Linux 的基本网络配置

17.1　基本网络配置的内容

通过对 Linux 主机进行基本的网络配置，可以使该主机能够同其他主机进行正常的通信。在进行基本网络配置之前，需要先掌握几个与网络相关的概念。

17.1.1　主机名

主机名用于标识一台主机的名称，通常该主机名在网络中是唯一的。如果该主机在 DNS 服务器上进行了域名的注册，主机名与该主机的域名通常也是一样的。

17.1.2　IP 地址

一台主机要在网络中和其他主机进行通信，首先要具有唯一的 IP 地址。当然 IP 地址一定是设置在主机的某块网卡上的。一般的主机只有一块网卡，所以在这种情况下把设置主机的 IP 地址和设置网卡的 IP 地址等同起来。如果主机中有多块网卡，每个网卡都可以设置独立的 IP 地址，则该主机可以拥有多个 IP 地址。

IP 地址的设置通常包括一系列的设置项，除 IP 地址本身外，还包括子网掩码、网络地址和广播地址，其中 IP 地址和子网掩码是必须提供的，网络地址和广播地址可以由 IP 地址和子网掩码进行计算得到。

主机的 IP 地址设置正确后就可以和同网段的其他主机进行通信了，但是只能使用 IP 地址而不能使用主机名进行通信。

17.1.3　网关地址

主机的 IP 地址设置正确后可以和同网段的其他主机进行通信，但还不能与不同网段的主机进行通信。为了实现与不同网段的主机进行通信，需要设置网关地址，该网关地址一定是同网段主机的 IP 地址，任何与不同网段主机进行的通信都将通过网关进行。

正确设置网关地址后，主机就可以与其他网段的主机进行通信，也许可以和接入互联网的任何主机进行通信，当然前提是作为网关的主机能够负担起网关的职责。

17.1.4　DNS 服务器地址

仅仅正确设置了 IP 地址和网关地址还不能使用域名和其他主机进行通信。为了能够使用域名而不是 IP 地址来连接主机，需要指定至少一个 DNS 服务器的 IP 地址，所有的域名解析（域名与 IP 地址间的相互转换）任务都将由该 DNS 服务器来完成。

这样就可以使用域名和其他主机进行通信了，当然如果愿意继续使用 IP 地址连接其他主机也是完全可以的。

17.2　网络配置相关文件

对于网络配置的全部内容都可以在系统中找到相关的配置文件，正是由于这些配置文件对网络选项的配置，Linux 系统启动时才能够正确配置网络启动系统。

17.2.1　模块配置文件

模块配置文件用于在 Linux 系统启动时加载系统所需的硬件驱动模块，如网卡和声卡等驱动模块，没有该文件对网卡驱动模块的配置，网卡将不能正常驱动，更谈不上其他配置了。

在 Red Hat Linux 中，模块配置文件的全路径名为/etc/modules.conf，通常情况下在 Linux 的安装过程中能够自动查询系统中的网卡并在 modules.conf 文件中进行设置，根据主机网卡硬件的不同，所加载的网卡驱动模块也会不同。如果系统中有多块网卡，需要对每块网卡都加载相应的驱动模块。

那么，系统中究竟加载了多少块网卡呢？在/etc/modules.conf 文件中会使用 alias 命令来标识。

```
#grep  eth  /etc/modules.conf
```

17.2.2　网卡 IP 地址配置文件

网卡 IP 地址配置文件位于目录/etc/sysconfig/network-scripts/中，文件名以“ifcfg-”开头，后跟网卡的类型（通常以太网用 eth 代表）加网卡序号（从 0 开始）。所示系统中以太网卡的配置文件名为“ifcfg-eth*N*”，其中 *N* 为从 0 开始的数字，如第一块以太网卡的配置文件名为 ifcfg-eth0，第二块以太网卡的配置文件名为 ifcfg-eth1，其他以此类推。

Linux 支持在一块物理网卡上绑定多个 IP 地址，需要建立多个网卡配置文件，其文件名形式为“ifcfg-eth*N*: *M*”，其中 *N* 和 *M* 都是相应的序号数字，如第一块以太网卡上的第一个虚拟网卡的配置文件名为 ifcfg-eth0:0。

所有的网卡 IP 地址配置文件都具有如下的类似格式，配置文件中每行进行一项内容设置，左边为项目名称，右边为项目设置值，中间用“=”分隔。配置文件中各项目的含义如表 17.1 所示。

表 17.1　网卡 IP 地址配置文件中各项的含义

项　目	设置值	说　　明
DEVICE	eth0	设备名称，第一块以太网卡为 eth0
ONBOOT	yes 或 no	设置系统启动时是否启动该设备
BOOTPROTO	none	启动协议，使用设置的 IP 地址时选择 none
IPADDR	192.168.1.19	该设备的 IP 地址
NETMASK	255.255.255.0	该设备的子网掩码
BROADCAST	192.168.1.255	广播地址，可以由 IP 地址和子网掩码计算而得
NETWORK	192.168.1.0	网络地址，可以由 IP 地址和子网掩码计算而得
GATEWAY	192.168.1.254	网关地址

```
[root@xinya ~]# cat /etc/sysconfig/network-scripts/ifcfg-eth0
# Advanced Micro Devices [AMD] 79c970 [PCnet32 LANCE]
DEVICE=eth0
BOOTPROTO=dhcp
HWADDR=00:0C:29:C5:1E:76
ONBOOT=yes
DHCP_HOSTNAME=xinya
```

17.2.3　DNS 客户配置文件

DNS 客户配置文件的全路径名为/etc/resolv.conf，在该文件中使用 nameserver 命令指定系统所使用的 DNS 服务器的 IP 地址，可以指定 3 个有效的 DNS 服务器地址。

除了指定 DNS 服务器外，resolv.conf 文件中还可以使用 domain 命令设置当前主机所在的域。

```
[root@xinya ~]# cat /etc/resolv.conf
; generated by /sbin/dhclient-script
nameserver 219.141.136.10
nameserver 219.141.140.10
```

17.2.4　名称解析顺序

在 UNIX 中除了 DNS 外还可以使用 nis、hosts 文件等方式进行名称解析，在同时使用多种方法进行名称解析时需要设定一个顺序，按照该顺序依次使用各种方法进行解析。

文件/etc/nsswitch.conf 中使用 hosts 关键字进行了名称解析顺序的设置，如下所示。

```
[root@xinya ~]# cat /etc/nsswitch.conf
…
hosts:      files dns
aliases:    files nisplus
…
```

hosts 后面的设置为解析顺序，当前的解析顺序为 files（表示 hosts 文件）、DNS 服务器和 NIS 服务器。在设置的解析序列中，可以设置全部的解析方法，也可以只使用其中的几种方法。通常只使用 hosts 文件和 DNS 进行解析。

17.2.5 hosts 文件

hosts 文件作为名称解析的一种方法，进行名称解析时系统直接读取该文件中设置的 IP 地址和主机名的对应记录。文件中每行为一个记录，IP 地址在左，主机名在右，主机名部分可以设置主机名和主机全域名。

```
[root@xinya ~]# cat /etc/hosts
# Do not remove the following line, or various programs
# that require network functionality will fail.
127.0.0.1           xinya localhost.localdomain localhost
::1                 localhost6.localdomain6 localhost6
```

该文件默认只有一条记录。127.0.0.1 是本机的环回地址，对应的是本机的名称。

17.3 网络相关命令

Linux 中提供了丰富的网络命令，有些命令用于配置网络，有些命令用于测试网络，而且许多命令都具有多种命令格式。熟练地掌握这些命令对于在 Linux 中配置和使用网络是大有益处的。

17.3.1 hostname 命令

1. 显示系统主机名

可以使用 hostname 命令显示当前系统主机名。

hostname

```
[root@xinya ~]# hostname
xinya
```

当前系统的主机名为 xinya。

2. 设置系统主机名

可以使用 hostname 命令设置系统主机名为指定的名称。

hostname 主机名称

如设置当前主机名为 hero：

```
[root@xinya ~]# hostname hero
[root@xinya ~]# hostname
hero
```

17.3.2 ifconfig

1. 显示当前活动的网卡设置

可以使用 ifconfig 命令查看当前系统中已启动的网卡。

ifconfig

```
[root@xinya ~]# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0C:29:C5:1E:76
          inet addr:192.168.1.109  Bcast:255.255.255.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fec5:1e76/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:3602 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2036 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:323240 (315.6 KiB)  TX bytes:267113 (260.8 KiB)
          Interrupt:169 Base address:0x2000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:1223 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1223 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:2077546 (1.9 MiB)  TX bytes:2077546 (1.9 MiB)
```

当前系统中的活动网卡包括 eth0 和 lo 两块，up 代表其状态为活动状态。这里的 lo 是环回地址网卡，用于测试本地协议栈的通信功能。

2. 显示系统中所有的网卡设置

当需要查看系统中的所有网卡信息，包括所有启动的和未启动的网卡，可以使用带有-a 选项的 ifconfig 命令。

ifconfig -a

```
[root@xinya ~]# ifconfig -a
eth0      Link encap:Ethernet  HWaddr 00:0C:29:C5:1E:76
          inet addr:192.168.1.109  Bcast:255.255.255.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fec5:1e76/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:3632 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2070 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
```

```
        RX bytes:326141 (318.4 KiB)  TX bytes:271597 (265.2 KiB)
        Interrupt:169 Base address:0x2000

lo      Link encap:Local Loopback
        inet addr:127.0.0.1  Mask:255.0.0.0
        inet6 addr: ::1/128 Scope:Host
        UP LOOPBACK RUNNING  MTU:16436  Metric:1
        RX packets:1223 errors:0 dropped:0 overruns:0 frame:0
        TX packets:1223 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:0
        RX bytes:2077546 (1.9 MiB)  TX bytes:2077546 (1.9 MiB)

sit0    Link encap:IPv6-in-IPv4
        NOARP  MTU:1480  Metric:1
        RX packets:0 errors:0 dropped:0 overruns:0 frame:0
        TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:0
        RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
```

3. 显示指定网卡的设置

“ifconfig”和“ifconfig -a”这两个命令用于显示一系列网卡设置，当需要查看特定网卡的设置时，需要在 ifconfig 命令中指定网卡名称。

ifconfig 网卡设备名称

如需要查看 eth0 网卡的详细设置：

```
[root@xinya ~]# ifconfig eth0
eth0    Link encap:Ethernet  HWaddr 00:0C:29:C5:1E:76
        inet addr:192.168.1.109  Bcast:255.255.255.255  Mask:255.255.255.0
        inet6 addr: fe80::20c:29ff:fec5:1e76/64 Scope:Link
        UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
        RX packets:3669 errors:0 dropped:0 overruns:0 frame:0
        TX packets:2112 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:329083 (321.3 KiB)  TX bytes:277185 (270.6 KiB)
        Interrupt:169 Base address:0x2000
```

4. 启动指定的网卡

主机中的网卡有启动和关闭两种状态，对于被关闭的非活动网卡而言，可以在 ifconfig 命令中加上 up 参数进行启动。

ifconfig 网卡名 up

```
[root@xinya ~]# ifconfig eth0 up
```

```
[root@xinya ~]# ifconfig eth0
eth0     Link encap:Ethernet  HWaddr 00:0C:29:C5:1E:76
         inet addr:192.168.1.109  Bcast:255.255.255.255  Mask:255.255.255.0
         inet6 addr: fe80::20c:29ff:fec5:1e76/64 Scope:Link
         UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
         RX packets:3738 errors:0 dropped:0 overruns:0 frame:0
         TX packets:2164 errors:0 dropped:0 overruns:0 carrier:0
         collisions:0 txqueuelen:1000
         RX bytes:334847 (326.9 KiB)  TX bytes:283196 (276.5 KiB)
         Interrupt:169 Base address:0x2000
```

5. 停止指定的网卡

对于已启动的活动网卡而言，可以使用 ifconfig 的 down 参数将网卡设置为关闭的非活动状态。

ifconfig　网卡名　down

```
[root@xinya ~]# ifconfig eth0
eth0     Link encap:Ethernet  HWaddr 00:0C:29:C5:1E:76
         inet addr:192.168.1.109  Bcast:255.255.255.255  Mask:255.255.255.0
         inet6 addr: fe80::20c:29ff:fec5:1e76/64 Scope:Link
         UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
         RX packets:3738 errors:0 dropped:0 overruns:0 frame:0
         TX packets:2164 errors:0 dropped:0 overruns:0 carrier:0
         collisions:0 txqueuelen:1000
         RX bytes:334847 (326.9 KiB)  TX bytes:283196 (276.5 KiB)
         Interrupt:169 Base address:0x2000
[root@xinya ~]# ifconfig eth0 down
[root@xinya ~]# ifconfig eth0
eth0     Link encap:Ethernet  HWaddr 00:0C:29:C5:1E:76
         inet addr:192.168.1.109  Bcast:255.255.255.255  Mask:255.255.255.0
         BROADCAST MULTICAST MTU:1500 METRIC:1
         RX packets:3738 errors:0 dropped:0 overruns:0 frame:0
         TX packets:2164 errors:0 dropped:0 overruns:0 carrier:0
         collisions:0 txqueuelen:1000
         RX bytes:334847 (326.9 KiB)  TX bytes:283196 (276.5 KiB)
          Interrupt:169 Base address:0x2000
```

6. 设置网卡的 IP 地址

为网卡设置 IP 地址是很重要的管理工作。设置网卡的 IP 地址可以使用 ifconfig 命令。

ifconfig　网卡名　IP 地址 [netmask 子网掩码]

例如，将网卡 eth0 的 IP 地址修改为 192.168.1.109：

```
[root@xinya ~]# ifconfig eth0
eth0     Link encap:Ethernet  HWaddr 00:0C:29:C5:1E:76
         inet addr:192.168.1.110  Bcast:255.255.255.255  Mask:255.255.255.0
         inet6 addr: fe80::20c:29ff:fec5:1e76/64 Scope:Link
         UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
         RX packets:3738 errors:0 dropped:0 overruns:0 frame:0
         TX packets:2164 errors:0 dropped:0 overruns:0 carrier:0
         collisions:0 txqueuelen:1000
         RX bytes:334847 (326.9 KiB)  TX bytes:283196 (276.5 KiB)
         Interrupt:169 Base address:0x2000
[root@xinya ~]# ifconfig eth0 192.168.1.109
[root@xinya ~]# ifconfig eth0
eth0     Link encap:Ethernet  HWaddr 00:0C:29:C5:1E:76
         inet addr:192.168.1.109  Bcast:255.255.255.255  Mask:255.255.255.0
         inet6 addr: fe80::20c:29ff:fec5:1e76/64 Scope:Link
         UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
         RX packets:3738 errors:0 dropped:0 overruns:0 frame:0
         TX packets:2164 errors:0 dropped:0 overruns:0 carrier:0
         collisions:0 txqueuelen:1000
         RX bytes:334847 (326.9 KiB)  TX bytes:283196 (276.5 KiB)
         Interrupt:169 Base address:0x2000
```

如果未使用 netmask 选项设置子网掩码，则子网掩码将为标准类型，本例中为 255.255.255.0。

17.3.3 ifup 命令

ifup 命令用于启动指定的非活动网卡设备，该命令与 ifconfig up 命令功能类似。

ifup　网卡设备名

例如，启动 lo 网络接口：

```
[root@xinya ~]# ifconfig lo
lo       Link encap:Local Loopback
         inet addr:127.0.0.1  Mask:255.0.0.0
         LOOPBACK  MTU:16436  Metric:1
         RX packets:1223 errors:0 dropped:0 overruns:0 frame:0
         TX packets:1223 errors:0 dropped:0 overruns:0 carrier:0
         collisions:0 txqueuelen:0
         RX bytes:2077546 (1.9 MiB)  TX bytes:2077546 (1.9 MiB)
[root@xinya ~]# ifup lo
[root@xinya ~]# ifconfig lo
lo       Link encap:Local Loopback
         inet addr:127.0.0.1  Mask:255.0.0.0
         inet6 addr: ::1/128 Scope:Host
         UP LOOPBACK RUNNING  MTU:16436  Metric:1
         RX packets:1223 errors:0 dropped:0 overruns:0 frame:0
```

```
        TX packets:1223 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:0
        RX bytes:2077546 (1.9 MiB)  TX bytes:2077546 (1.9 MiB)
```

这时 lo 已处于活动状态。

17.3.4 ifdown 命令

ifdown 命令用于停止指定的活动网卡设备，该命令与 ifconfig down 命令功能类似。

ifdown 网卡设备

```
[root@xinya ~]# ifconfig lo
lo      Link encap:Local Loopback
        inet addr:127.0.0.1  Mask:255.0.0.0
        inet6 addr: ::1/128 Scope:Host
        UP LOOPBACK RUNNING  MTU:16436  Metric:1
        RX packets:1223 errors:0 dropped:0 overruns:0 frame:0
        TX packets:1223 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:0
        RX bytes:2077546 (1.9 MiB)  TX bytes:2077546 (1.9 MiB)
[root@xinya ~]# ifdown lo
[root@xinya ~]# ifconfig lo
lo      Link encap:Local Loopback
        LOOPBACK  MTU:16436  Metric:1
        RX packets:1223 errors:0 dropped:0 overruns:0 frame:0
        TX packets:1223 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:0
        RX bytes:2077546 (1.9 MiB)  TX bytes:2077546 (1.9 MiB)
```

这时 lo 处于非活动状态。

17.3.5 route 命令

route 命令用于显示和动态修改系统当前的路由表信息。该路由表在系统运行时始终有效，系统一旦重新启动，网络将丢失使用 route 命令设置的路由信息。

1. 显示路由信息

当需要显示系统的路由信息时，可以使用 route 命令。该命令在不添加任何参数时将显示当前的路由信息。

route

```
[root@xinya ~]# route
Kernel IP routing table
Destination    Gateway   Genmask         Flags  Metric  Ref  Use  Iface
192.168.1.0    *         255.255.255.0   U      0       0    0    eth0
169.254.0.0    *         255.255.0.0     U      0       0    0    eth0
```

2. 添加和删除路由信息

当需要在系统的路由表中添加或删除路由信息时，可以使用 route 命令的 add 或 del 参数。

以下命令用于在路由表中添加路由条目。

route add -net 网络地址 netmask 子网掩码 dev 网卡设备名

例如，在本地主机的路由表中添加通过 eth0 去往 10.0.0.0 网络的路由：

```
[root@xinya ~]# route add -net 10.0.0.0 netmask 255.0.0.0 dev eth0
[root@xinya ~]# route
Kernel IP routing table
Destination   Gateway    Genmask         Flags   Metric  Ref  Use  Iface
192.168.1.0   *          255.255.255.0   U       0       0    0    eth0
169.254.0.0   *          255.255.0.0     U       0       0    0    eth0
10.0.0.0      *          255.0.0.0       U       0       0    0    eth0
```

以下命令用于从路由表中删除路由条目。

route del -net 网络地址 netmask 子网掩码

如在本地主机的路由表中删除去往 10.0.0.0 网络的路由：

```
[root@xinya ~]# route
Kernel IP routing table
Destination   Gateway    Genmask         Flags  Metric  Ref  Use  Iface
192.168.1.0   *          255.255.255.0   U      0       0    0    eth0
10.0.0.0      *          255.0.0.0       U      0       0    0    eth0
[root@xinya ~]# route  del  -net  10.0.0.0  netmask  255.0.0.0
[root@xinya ~]# route
Kernel IP routing table
Destination   Gateway    Genmask         Flags  Metric  Ref  Use  Iface
192.168.1.0   *          255.255.255.0   U      0       0    0    eth0
```

3. 添加和删除默认网关

可以添加与删除本地主机路由表中的默认网关记录。

以下命令用于向路由表中添加默认网关地址。

Route add default gw 网关 IP 地址 dev 网卡设备名称

如将本地主机的默认网关设置为 192.168.1.1：

```
[root@xinya ~]# route add default gw 192.168.1.1 dev eth0
Kernel IP routing table
Destination   Gateway      Genmask          Flags Metric Ref  Use  Iface
```

```
192.168.1.0     *             255.255.255.0   U     0      0    0   eth0
169.254.0.0     *             255.255.0.0     U     0      0    0   eth0
default         192.168.1.1   0.0.0.0         UG    0      0    0   eth0
```

以下命令用于从路由表中删除默认网关地址。

```
Route del default gw 网关 IP 地址
```

4. ping

ping 命令是常用的网络测试命令，该命令通过向被测试的目的主机地址发送 ICMP 报文并收取回应报文来测试当前主机到目的主机的网络连接状态。

ping 命令默认会不间断地发送 ICMP 报文，直到用户终止该命令，使用-c 参数并指定相应的数目，可以控制 ping 命令发送报文的数量。

例如，判断当前主机同主机 192.168.1.104 之间的网络连接状态：

```
[root@xinya ~]# ping 192.168.1.104
PING 192.168.1.104 (192.168.1.104) 56(84) bytes of data.
64 bytes from 192.168.1.104: icmp_seq=1 ttl=64 time=2.71 ms
64 bytes from 192.168.1.104: icmp_seq=2 ttl=64 time=0.964 ms
64 bytes from 192.168.1.104: icmp_seq=3 ttl=64 time=1.72 ms
64 bytes from 192.168.1.104: icmp_seq=4 ttl=64 time=0.870 ms
64 bytes from 192.168.1.104: icmp_seq=5 ttl=64 time=0.844 ms
64 bytes from 192.168.1.104: icmp_seq=6 ttl=64 time=0.799 ms

--- 192.168.1.104 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5003ms
rtt min/avg/max/mdev = 0.799/1.319/2.712/0.699 ms
```

可以使用 Ctrl+C 组合键来中止 ping 命令的运行。

也可以使用“-c *n*”选项指定 ping 命令发送 ICMP 报文的数量，即发送多少个 ICMP 报文后自动退出 ping，*n* 为发送 ICMP 报文的次数。例如：

```
[root@xinya ~]# ping -c 4 192.168.1.104
PING 192.168.1.104 (192.168.1.104) 56(84) bytes of data.
64 bytes from 192.168.1.104: icmp_seq=1 ttl=64 time=0.860 ms
64 bytes from 192.168.1.104: icmp_seq=2 ttl=64 time=0.956 ms
64 bytes from 192.168.1.104: icmp_seq=3 ttl=64 time=0.925 ms
64 bytes from 192.168.1.104: icmp_seq=4 ttl=64 time=0.882 ms

--- 192.168.1.104 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3001ms
rtt min/avg/max/mdev = 0.860/0.905/0.956/0.052 ms
```

本章对 Linux 操作系统的基本网络配置进行了介绍。网络配置是网络应用的基础，只有事先准备好基本的网络环境，才能在此基础上搭建各种网络应用服务。

第3篇　Shell 基础

Shell 在 Linux 操作系统管理中有着特殊的地位，它在实现自动管理操作方面有着不可替代的作用。但是确实需要下很大的功夫才能很好掌握 Shell。有关 Shell 的内容很多，甚至可以为“Shell 脚本”单独开设一门课程。本书用两章对 Shell 脚本程序设计做概括性的介绍，力求使读者尽快认识 Shell 及其在系统管理中的作用。

第18章 Shell Script 基础

Shell Script 是 Linux 系统管理工作中非常重要的一种管理工具，在处理重复性操作、自动管理等相关工作方面有着灵活、稳定的特点，因此 Shell 脚本在系统管理工作中是经常使用的。

关于 Shell 脚本的知识将分为两部分来介绍，本章介绍 Shell 脚本的基础知识，下一章介绍 Shell 脚本的结构控制语句。

18.1 简单的 Shell 脚本

Shell Script 是利用 Shell 功能所编写的，用于按照既定目的进行处理的一种解释型程序。掌握 Shell 脚本有利于提高系统的自动化管理能力和对数据处理的效率。

为了更快地认识 Shell 脚本，首先来编写第一个 Shell 脚本 script01.sh。

```
[root@xinya script]# vi script01.sh  ←创建 script01.sh 文件并编辑
```

以下是文件内容。

```
#!/bin/bash
# Program: This program shows "Hello world!" in your screen.

echo -e  "Hello world!\a\n"
exit 0
```

文件编辑完成后，保存文件并退出，然后修改文件权限，为文件添加可执行权限。

```
[root@xinya script]# ll
-rw-r--r-- 1 root root 110 05-15 18:43 script01.sh
[root@xinya script]# chmod +x script01.sh
[root@xinya script]# ll
总计 4
-rwxr-xr-x 1 root root 110 05-15 18:43 script01.sh
```

权限修改完成后，执行“./脚本文件名”命令，如输入“./script01.sh”即执行当前目录下的脚本文件 script01.sh。

```
[root@xinya script]# ./script01.sh
Hello world!
```

这就是我们所写的第一个简单的脚本。脚本虽然简单，但是它反映出脚本在编写和执行时的步骤和注意事项。

18.1.1　Shell 脚本编写的约定

（1）Shell 脚本必须以“#!/bin/bash”这个固定格式开头，且该信息应位于脚本文件的第一行。

（2）脚本中命令的执行是自上而下、从左到右分析执行的。

（3）脚本中以#号开头的行是注释信息，不会对脚本的运行产生影响。

（4）脚本中的空白行将被忽略。

18.1.2　Shell 脚本的执行方法

在编辑好脚本之后，需要为脚本添加可执行权限，以使脚本具有调用/bin/bash 程序的能力。若没有为脚本添加可执行权限，也可以直接使用 bash 执行脚本，其使用方法是：

bash 脚本名

```
[root@xinya script]# bash script01.sh
Hello world!
```

由于脚本所在的目录不一定位于$PATH 变量中，所以在执行脚本时应使用绝对路径。若需要执行当前目录下的脚本，可以使用“./脚本名”这种格式执行。

若脚本集中存储于某个目录中，可将该目录的绝对路径添加至 PATH 环境变量，如此一来，就可以在忽略路径信息的情况下直接利用脚本文件名来执行脚本了。

在 Linux 系统中，脚本文件常使用“.sh”作为文件的扩展名。脚本的执行是在当前 Shell 下新建一个子 shell 来执行的。若需要让脚本在当前 Shell 下执行，则可以使用“source 脚本文件名”或“.脚本文件名”命令来执行脚本。

18.1.3　脚本的基本结构

下面以刚刚编写的脚本 script01.sh 为例来介绍脚本的基本结构。

```
[root@xinya script]# cat script01.sh
#!/bin/bash
# Program: This program shows "Hello world!" in your screen.

echo -e  "Hello world!\a\n"
exit 0
```

该脚本虽然很简单，但具备了一个脚本所应具有的基本结构。下面介绍该脚本的基本结构。

（1）#!/bin/bash：是 bash 脚本的标准头部写法，bash 脚本必须以该字符串开头。其中“#!”用于定义该脚本文件使用哪种 Shell 程序来执行，这里 bash 脚本应使用/bin/bash 这个 Shell 程序来执行。

（2）#号开头的是脚本的注释内容，注释内容用于说明脚本信息。在编写脚本时，应利用注释信息明确脚本的目的、版本信息、作者和相关 Bug 信息，以便后续的脚本使用者能快速识别和接手该脚本的维护工作。

（3）脚本程序主体实际上就是一系列的命令和结构控制语句的集合。前面各章所介绍的命令均可以在脚本中应用，再结合下一章所介绍的结构控制语句，可以使脚本能自动完成很多以前只能由管理员手工完成的操作。

（4）“exit 0”是程序运行结果的返回码，若脚本成功执行，则其返回状态应在脚本的末尾进行定义，以便相关程序调用该脚本。

18.2　常见的 Shell 脚本要素

虽然从理论上讲 Shell 脚本是 Shell 命令的集合，但是在 Shell 脚本的编写中还是具有一些常用的命令和表达方式的。这些命令和表达方式我们在前面各章已经接触过了，本节将利用前面学习过的命令来创建几个简单的 Shell 脚本，以便更好地认识 Shell 脚本。

18.2.1　echo 命令的使用

echo 命令用于屏幕打印输出，其基本语法格式为：

【语法】**echo　[选项]　"字符串"**

选项说明如下：

-n：原样输出字符串，输出后不换行。

-e：原样输出字符串，支持在字符串中使用控制字符。常用的控制字符见表 18.1。

表 18.1　常用的控制字符

控制字符	含　　义	控制字符	含　　义
\a	发出警告声	\n	换行
\b	删除前一个字符	\r	光标移至行首
\c	最后不加换行符	\t	插入 Tab 制表符

例如：

```
[root@xinya script]# echo -e "*\t*\t*\t"
*       *       *   ←注意,输出字符串中添加了制表符
```

18.2.2　利用 read 命令实现脚本的交互式操作

read 命令用于由键盘读取输入，并将其复制给变量。在脚本中使用 read 命令可以实现脚本与用户交互的操作。

【示例】编写一个脚本，要求用户输入 Firstname 和 Lastname，并输出“Hello，用户名，welcome to 主机名”。

```
[root@xinya script]# vi script02.sh
```

```
#!/bin/bash
#program : Input name and Output
#           "Hello,name"
#           "Welcome to hostname"

read -p "Input your Firstname :" name1
read -p "Input your Lastname :" name2
echo -e "Hello,$name1.$name2\nWelcome to $HOSTNAME"
exit 0
[root@xinya script]# chmod +x script02.sh
[root@xinya script]# ./script02.sh
Input your Firstname : zhang
Input your Lastname : san
Hello,zhang.san
Welcome to xinya
```

在上例的脚本中使用了 read 命令，实现了脚本与用户之间的交互。同时，该脚本还引用了变量，如此一来，脚本的功能变得更灵活和丰富了。

18.2.3　脚本中为变量赋值的操作

脚本中的变量赋值操作与命令行相同。

【示例】利用日期为文件命名，文件名格式为“file-年月日”，日期取前天、昨天和今天。

```
#!/bin/bash
read -p "Enter the first file name:" fileuser
date1='date --date='2 days ago' +%Y%m%d'
date2='date --date='1 days ago' +%Y%m%d'
date3='date +%Y%m%d'
file1="$fileuser-$date1"
file2="$fileuser-$date2"
file3="$fileuser-$date3"
touch $file1 $file2 $file3
ls -l $file1 $file2 $file3
[root@xinya script]#chmod +x script03.sh
[root@xinya script]#./script03.sh
Enter the first file name:test
-rw-r--r-- 1 root root 0 May 17 10:03 test-20110515
-rw-r--r-- 1 root root 0 May 17 10:03 test-20110516
-rw-r--r-- 1 root root 0 May 17 10:03 test-20110517
```

18.3　脚本中的判断命令

在脚本中经常会涉及对文件的判断，包括文件是否存在、文件权限是否可读等操作。同时还存在数值比较和字符串比较等操作。这些比较操作均可以通过 test 命令来实现。

18.3.1　利用 test 命令进行文件判断

test 命令可以对文件进行判断，包括文件是否存在、文件是否为特定属性文件等。

【示例】判断当前目录下是否存在文件 abc。若文件不存在，则创建该文件；若文件已存在，则显示“File already exists”。

判断文件是否存在可以使用“test -e 文件名”命令，该命令用于判断文件是否存在，若文件存在，则命令状态返回值为“0”，否则返回值不为“0”。

```
[root@xinya script]#ls
script03.sh
[root@xinya script]#test -e abc && echo "File already exists" || touch abc
[root@xinya script]#ls
abc  script03.sh
[root@xinya script]#test -e abc && echo "File already exists" || touch abc
File already exists
```

用于进行文件判断的 test 命令见表 18.2。

表 18.2　用于文件判断的 test 命令

命　　令	作　　用
test -e　文件名	判断文件名是否存在
test -f　文件名	判断文件名是否存在且为文件
test -d　文件名	判断文件名是否存在且为目录
test -b　文件名	判断文件名是否存在且为块设备文件
test -c　文件名	判断文件名是否存在且为字符设备文件
test -S　文件名	判断文件名是否存在且为 Socket 文件
test -p　文件名	判断文件名是否存在且为 FIFO 管道文件
test -L　文件名	判断文件名是否存在且为链接文件

18.3.2　利用 test 命令进行文件权限判断

文件权限判断包括文件是否可读、可写及可执行等。

【示例】判断当前目录下的文件 abc 是否为可执行文件。若是，则显示文件为可执行文件，否则为文件添加可执行权限。

判断文件是否为可执行文件使用“test -x 文件名”命令，若文件为可执行文件，则命令状态返回值为 0，否则返回值不为 0。

```
[root@xinya script]#ll
total 4
-rw-r--r-- 1 root root   0 May 17 10:20 abc
-rwxr-xr-x 1 root root 290 May 17 10:03 script03.sh
```

```
[root@xinya script]#test -x abc && echo "file abc is executable file" ||
chmod +x abc
[root@xinya script]#ls -l
total 4
-rwxr-xr-x 1 root root   0 May 17 10:20 abc
-rwxr-xr-x 1 root root 290 May 17 10:03 script03.sh
[root@xinya script]#test -x abc && echo "file abc is executable file" ||
chmod +x abc
file abc is executable file
```

用于进行文件权限判断的 test 命令见表 18.3。

表 18.3　用于文件权限判断的 test 命令

命　令	作　　用
test -r　文件名	判断文件名是否存在，且用户对文件具有可读权限
test -w　文件名	判断文件名是否存在，且用户对文件具有可写权限
test -x　文件名	判断文件名是否存在，且用户对文件具有可执行权限
test -u　文件名	判断文件名是否存在，且文件具有 SUID 权限
test -g　文件名	判断文件名是否存在，且文件具有 SGID 权限
test -k　文件名	判断文件名是否存在，且文件具有 Sticky bit 权限
test -s　文件名	判断文件名是否存在，且为非空白文件

18.3.3　利用 test 命令比较文件新旧

当需要判断两个文件中哪一个比较新时，test 命令提供了比较新旧的方法。

【示例】判断当前目录下的 script03.sh 和 abc 这两个文件哪个比较新。

判断文件的新旧可以使用“test　file1 -nt　file2”命令，该命令用于判断 file1 是否比 file2 新，若是则命令状态返回值为 0，否则返回值为非 0。

```
[root@xinya script]#ll
total 4
-rwxr-xr-x 1 root root   0 May 17 10:20 abc
-rwxr-xr-x 1 root root 290 May 17 10:03 script03.sh
[root@xinya script]#test abc -nt script03.sh && echo "file abc is new" ||
echo "file srcipt03.sh is new"
file abc is new
[root@xinya script]#touch script03.sh
[root@xinya script]#test abc -nt script03.sh && echo "file abc is new" ||
echo "file srcipt03.sh is new"
file srcipt03.sh is new
```

用于进行文件新旧比较的 test 命令见表 18.4。

表 18.4 用于文件新旧比较的 test 命令

命　令	作　　用
test file1 -nt file2	判断 file1 是否比 file2 新
test file1 -ot file2	判断 file1 是否比 file2 旧
test file1 -ef file2	判断 file1 与 file2 是否为同一文件（硬链接）

18.3.4 利用 test 命令进行数值比较

尽管用户可以直接判断两个数值的大小关系，但是对于计算机而言还是需要一个特定的语法结构用于比较两个数值间的大小关系，特别是数值以变量的形式存在的情况下，更是需要这种判断。

test 命令提供的判断数值的命令主要是用于判断两个数值的大小与是否相等。

【示例】判断两个整型变量$a 与$b 的值的大小关系。

判断两个数值大小可以使用“test n1 -gt n2”命令，该命令用于判断数值 n1 是否大于 n2，若结果为真则命令状态返回值为 0，否则为非 0。

```
[root@xinya script]#a=$RANDOM
[root@xinya script]#b=$RANDOM
[root@xinya script]#test $a -gt $b && echo 'a>b' || echo 'a<b'
a>b
[root@xinya script]#echo "$a $b"
11173 8153
```

用于判断两个数值的大小关系的 test 命令见表 18.5。

表 18.5 用于判断两个数值的大小关系的 test 命令

命　　令	作　　用
test n1 -eq n2	判断两数值是否相等
test n1 -ne n2	判断两数值是否不相等
test n1 -gt n2	判断 n1 是否大于 n2（n1>n2）
test n1 -ge n2	判断 n1 是否大于等于 n2（n1≥n2）
test n1 -lt n2	判断 n1 是否小于 n2（n1<n2）
test n1 -le n2	判断 n1 是否小于等于 n2（n1≤n2）

18.3.5 利用 test 命令进行字符串判断

字符串判断主要是判断是否为空以及两个字符串是否相等。

【示例】判断两个字符串的内容是否相等。

判断两个字符串是否相等可以使用“test 字符串 1=字符串 2”命令。

```
[root@xinya script]#a='Hello'
```

```
[root@xinya script]#b='welcome'
[root@xinya script]#test $a = $b;echo $?
1
```

【示例】判断 user1 与 user2 是否使用相同的登录 Shell。

```
[root@xinya script]#vi script04.sh
#!/bin/bash
read -p "Enter the first username:" user1
read -p "Enter the second username:" user2
a='grep "^$user1" /etc/passwd | cut -d : -f 7'
b='grep "^$user2" /etc/passwd | cut -d : -f 7'
test $a = $b && echo "login shell is $a" || echo "$user1 login shell is
$a,$user2 login shell is $b"
[root@xinya script]#chmod +x script04.sh
[root@xinya script]#./script04.sh
Enter the first username:user1
Enter the second username:user2
login shell is /bin/sh
[root@xinya script]#tail -n 2 /etc/passwd
user1:x:1001:1003::/home/user1:/bin/sh
user2:x:1002:1004::/home/user2:/bin/sh
[root@xinya script]#./script04.sh
Enter the first username:user1
Enter the second username:user2
user1 login shell is /bin/ksh,user2 login shell is /bin/sh
```

用于进行字符串判断的 test 命令见表 18.6。

表 18.6　用于字符串判断的 test 命令

命　令	作　　用
test -z　字符串	判断字符串是否为 0，若为 0 则为真
test -n　字符串	判断字符串是否不为 0，若不为 0 则为真
test 字符串 1=字符串 2	判断字符串 1 与字符串 2 是否相等，若相等则为真
test 字符串 1！=字符串 2	判断字符串 1 与字符串 2 是否不相等，若不相等则为真

18.3.6　test 命令的逻辑判断

test 命令提供了与、或、非 3 种逻辑判断。

【示例】判断文件 abc 是否为可读、可写与可执行文件。

test 提供了与条件判断，其语法结构为“test 判断 1 -a 判断 2”，若判断 1 与判断 2 同时成立，则结果为真，否则结果为假。

```
[root@xinya script]#ll
total 8
```

```
-rwxr-xr-x 1 root root   0 May 17 10:20 abc
-rwxr-xr-x 1 root root 290 May 17 10:47 script03.sh
-rwxr-xr-x 1 root root 293 May 17 11:33 script04.sh
[root@xinya script]#test -r abc -a -w abc -a -x abc;echo $?
0
```

用于进行逻辑判断的 test 命令见表 18.7。

表 18.7　用于逻辑判断的 test 命令

命　　令	作　　用
test 判断 1 -a 判断 2	与条件判断，判断 1 与判断 2 同时成立则结果为真，否则结果为假
test 判断 1 -o 判断 2	或条件判断，判断 1 或判断 2 中只要一个判断成立即为真
test ！判断 1	非条件判断，判断 1 的否定值结果为真

18.4　利用判断符号[ ]

在 Linux 系统中，除使用 test 命令进行判断外，还可以使用“[ ]”符号进行判断，“[ ]”判断符号的用法与 test 命令相同，是 test 命令的另一种表现形式。

【示例】判断两个字符串的内容是否相等。

使用“[ ]”符号判断两个字符串是否相等的语法格式为：

[字符串 1==字符串 2]

```
[root@xinya script]#a='Hello'
[root@xinya script]#b='welcome'
[root@xinya script]#[  "$a"  ==  "$b"  ] ;echo $?
1
```

在上例中，比较两个字符串时使用了==运算符，该运算符与=的作用是相同的，不过系统上在为变量赋值时使用=号，而在进行比较操作时常用的是==符号。

在使用“[ ]”符号进行判断操作时要注意，在判断符号“[ ]”中每个组件间均应使用空格进行分隔，对在判断符号“[ ]”中引用变量，建议使用" "来标注，如：

```
["$a"  ==  "$b"  ]
```

【示例】判断当前目录下是否存在 abc 文件，若存在则询问用户是否删除该文件，用户回答“y”则删除该文件，用户回答“n”则退出。若不存在 abc 文件，则询问用户是否创建该文件，用户回答“y”则创建该文件，用户回答“n”则退出。

```
[root@xinya script]# vi script04.sh
#!/bin/bash
read -p "Enter a file name :" filename
[ -f "$filename" ] && read -p "Delete the file $filename ? (y/n)"  an1 ‖
read -p "Create the file $filename ?(y/n)" an2
```

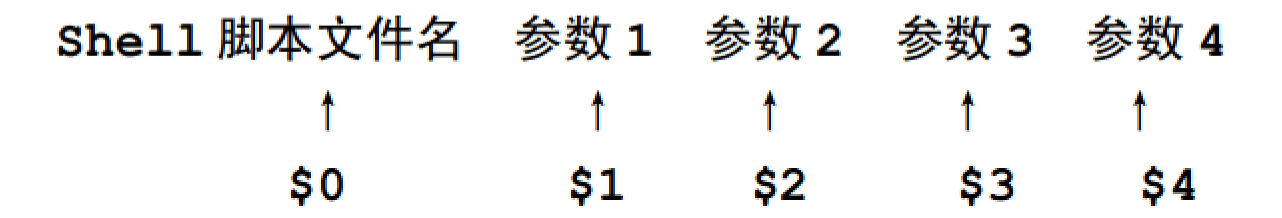
```
[  "$an1" == "y"  -o  "$an1"  ==  "Y"  ]  &&  rm -f  $filename
[  "$an2" == "y"  -o  "$an2"  ==  "Y"  ]  &&  touch  $filename

exit 0
[root@xinya script]# chmod +x script04.sh
[root@xinya script]# ls
script01.sh  script02.sh  script04.sh
[root@xinya script]# ./script04.sh
Enter a file name :abc
Create the file abc ?(y/n)y
[root@xinya script]# ls
abc  script01.sh  script02.sh  script04.sh
```

18.5　Shell 的默认变量

Shell 脚本也支持参数的调用，Shell 脚本的参数是通过默认的变量值来体现的。

Shell 脚本的参数变量结构如下：



脚本名在脚本中使用变量$0 来表示，脚本的第一个参数使用变量$1 来表示，依此类推。

【示例】输入网络接口、IP 地址、子网掩码和默认网关，自动设置主机的网络配置。

```
[root@xinya script]# vi script05.sh
#!/bin/bash
ifconfig $1 $2 netmask $3
route add default gw $4
ifconfig $1
[root@xinya script]# chmod +x script05.sh
[root@xinya script]# ./script05.sh  eth0:1 192.168.1.254 255.255.255.0
192.168.1.1
eth0:1    Link encap:Ethernet  HWaddr 00:0C:29:C5:1E:76
          inet addr:192.168.1.254 Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          Interrupt:169 Base address:0x2000
```

第 19 章　Shell Script 中的结构控制语句

在 Shell 脚本中除使用标准的 Linux 命令程序外，还可以利用结构控制语句来实现流程控制。常见的流程控制结构包括条件判断结构和循环结构。

19.1　条件判断语句

所谓条件判断是指定义一个既存条件，当条件结果值为真和为假时分别进行何种操作。下面分别介绍条件判断语句的两种语法结构。

19.1.1　if … then 判断语句

语法结构如下：

```
if  条件判断;
    then
        程序描述
fi
```

其语法结构图如图 19.1 所示。

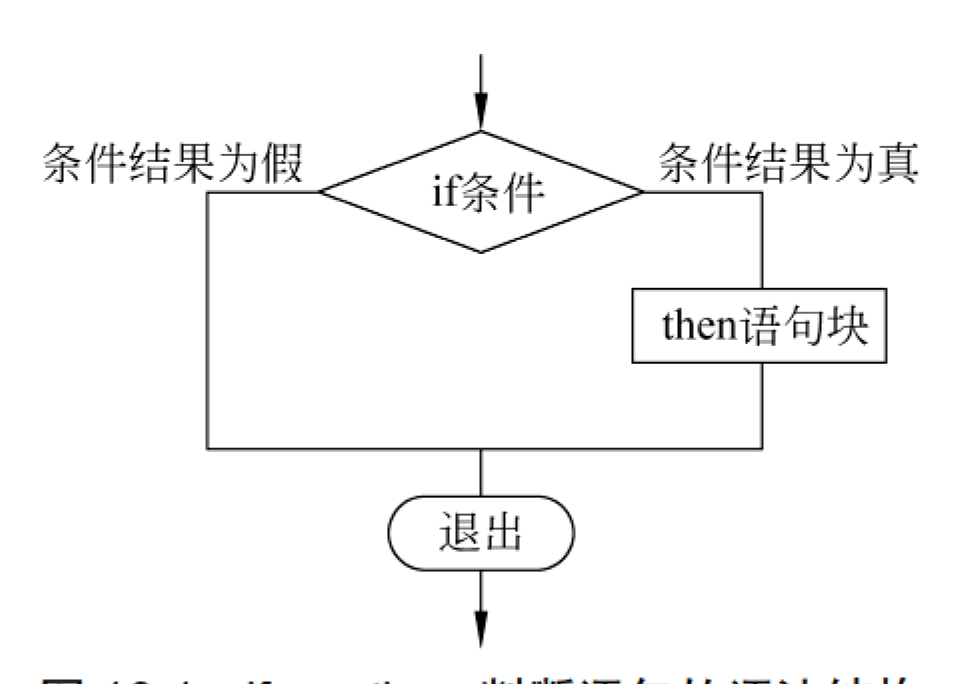


图 19.1　if … then 判断语句的语法结构

该语法结构的含义为：定义一个判断条件。当条件成立时（条件结果为真），就执行 then 后的语句块，执行完退出 if 判断；当条件不成立时（条件结果为假），直接退出 if 判断。

【示例】判断用户输入的文件是否存在。若存在则询问用户是否删除该文件，用户回答“y”则删除该文件，用户回答“n”则退出；若不存在 abc 文件，则询问用户是否创建该文件，用户回答“y”则创建该文件，用户回答“n”则退出。

这个示例在上一章出现过，但由于当时还没有学习结构控制语句，所以其实现过程显得很简单。此处使用 if 语句来实现其脚本功能。

```
[root@xinya script]# vi script06.sh
#!/bin/bash
read -p "Enter a filename : "  filename
[ -f "$filename" ] && read -p "Delete the file $filename (y/n)" an1 || read
-p "Create the file $filename (y/n)" an2
```

```
if ["$an1"=="y" -o "$an1"=="Y"];
        then
                rm -f $filename
                echo "$filename is delete"
fi

if ["$an1"=="n" -o "$an1"=="N"];
        then
                echo "$filename is not delete"
                exit 0
fi

if ["$an2"=="y" -o "$an2"=="Y"];
        then
                touch $filename
                ls -l $filename
fi

if ["$an2"=="n" -o "$an2"=="N"];
        then
                echo "$filename is not exist"
                exit 0
fi
exit 0

[root@xinya script]# chmod +x script06.sh
[root@xinya script]# ./script06.sh
Enter a filename : abc
Delete the file abc (y/n)y
abc is delete
[root@xinya script]# ./script06.sh
Enter a filename : abc
Create the file abc (y/n)y
-rw-r--r-- 1 root root 0 05-16 12:26 abc
```

注意，因为基于用户的回答有 4 种可能，所以脚本中使用了 4 个 if … then 语句来进行判断。

在使用 if 语句时要注意，在 if 语句的判断条件结尾需要使用“;”来定界，且 if 语句需要使用 fi 来表示语句的结束。

if … then 语句实现的是一种简单的判断。从上例中可见，该语句只能针对判断条件为真做出相应的处理，当条件不为真时则需要重新利用 if 判断。其实，if 语句还提供了更进一步的语法结构，用于在条件为真和条件为假时分别进行处理。

19.1.2 if … then … else 二重判断

所谓二重判断是指在 if 语句中，当定义的条件为真时执行 then 后定义的操作，当定义的条件为假时执行 else 后定义的操作，其语法结构为：

```
if  条件判断;
    then
        程序描述
    else
        程序描述
fi
```

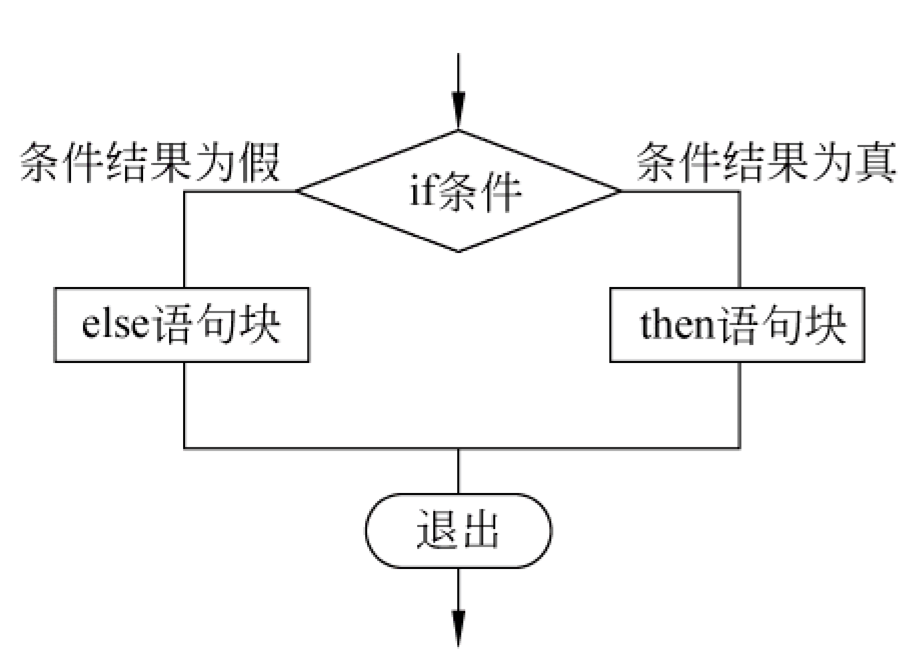


图 19.2 if … then … else 判断语句的语法结构

其语法结构图如图 19.2 所示。

此处使用 if … then … else 语法结构来实现上一个示例所要求的脚本功能，由于二重判断的存在，所以其实现比单独使用 if … then 要简洁许多。

```
[root@xinya script]# vi script07.sh
#!/bin/bash
read -p "Enter a filename : "  filename
[ -f "$filename" ] && read -p "Delete the file $filename (y/n)" an1 || read
-p "Create the file $filename (y/n)" an2

if ["$an1"=="y" -o "$an1"=="Y"];
       then
              rm -f $filename
              echo "$filename is deletei"
       else
              echo "$filename is not delete"
              exit 0
fi

if ["$an2"=="y" -o "$an2"=="Y"];
       then
              touch $filename
              ls -l $filename
       else
              echo "$filename is not exist"
              exit 0
fi
exit 0

[root@xinya script]# chmod +x script07.sh
[root@xinya script]# ./script07.sh
```

```
Enter a filename : abc
Delete the file abc (y/n)y
abc is deletei
[root@xinya script]# ./script07.sh
Enter a filename : abc
Create the file abc (y/n)y
abc is not delete
```

注意，这个实现过程比上一个实现过程要简单得多。但是在该实现过程中仅能判断用户输入“y”以后的操作，对用户是否输入“n”则无法判断。我们也可以对该脚本进一步完善，以确定用户输入“n”后的操作和输入“y”与“n”之外的字符的操作。

```
#!/bin/bash
read -p "Enter a filename : "  filename
[ -f "$filename" ] && read -p "Delete the file $filename (y/n)" an1 || read
-p "Create the file $filename (y/n)" an2

if ["$an1"=="y" -o "$an1"=="Y"];
      then
            rm -f $filename
            echo "$filename is deletei"
      else
            if ["$an1"=="n" -o "$an1"=="N"];
                  then
                        echo "$filename is not delete"
                        exit 0
                  else
                         if ["$an1"!="n" -a "$an1"!="y" -a "$an1"!=
                         "y"-a"$an1"!="Y"];
                               then

                                     echo "Input the correct option,please"
                                     exit 0
                         fi

             fi
fi

if ["$an2"=="y" -o "$an2"=="Y"];
      then
             touch $filename
             ls -l $filename
      else
             if  ["$an2"=="n" -o "$an2"=="N"];
                   then
```

```
                        echo "$filename is not exist"
                        exit 0
                else
                        if ["$an2"!="n" -a "$an2"!="y" -a "$an2"!="y"
                        -a "$an2"!= "Y"];
                              then

                                    echo "Input the correct option,please"
                                    exit 0
                        fi

            fi
fi
exit 0
```

该脚本使用了 if 的嵌套判断，用以实现当用户输入“y”、“n”和其他字符时脚本所做的响应。if 嵌套在多重条件判断时是经常应用的。

if 嵌套的另一种语法格式为：

```
if 判断条件 1;
    then
        程序描述
    elif 判断条件 2;
        then
            程序描述
        else
            程序描述
fi
```

19.2 循 环 语 句

if 是一种判断结构，在脚本中还存在一种循环结构。所谓循环结构是指定义一个既定条件，当条件成立时执行程序体，直到条件不成立时为止。

19.2.1 循环语句 while … do

while 语句的语法结构：

```
while [ 判断条件 ]
do
    程序描述
done
```

当判断条件成立时开始执行 do 命令所定义的程序描述，一个程序体执行完成后再做条件判断，若条件仍然成立，则继续执行 do 命令定义的程序体，直到判断条件不成立时

为止。

其语法结构图如图 19.3 所示。

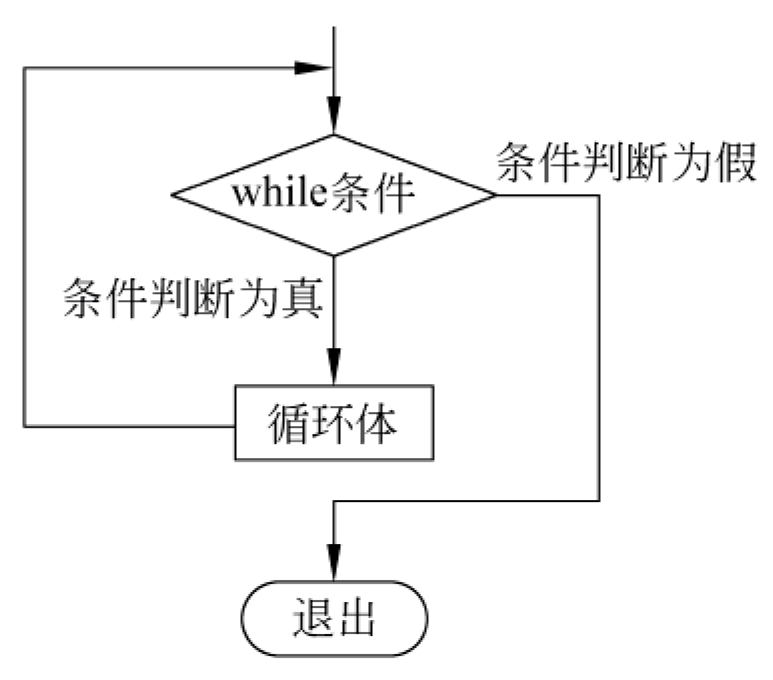


图 19.3　while … do 循环的语法结构

【示例】由 1 开始，输出 100 个数。

```
[root@xinya script]# vi script08.sh
#!/bin/bash
a=0
while [  "$a"  -le  '100' ]
do
       echo -n "$a  "
       a=$(( $a+1 ))
done
echo

[root@xinya script]# chmod +x script08.sh
[root@xinya script]# ./script08.sh
0  1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21
22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59
60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78
79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97
98  99  100
```

【示例】打印简单的三角形。

```
[root@xinya script]# vi script09.sh
#!/bin/bash
a=1
b=1
while [ "$a" -le "5" ]
do
       while [  "$b"  -le  "$a" ]
       do
              c='*'
              s="$s"$c
```

```
                b=$(( $b+1 ))
        done
        echo "$s"
        a=$(( $a+1 ))
done

[root@xinya script]# chmod +x script09.sh
[root@xinya script]# ./script09.sh
*
**
***
****
*****
```

19.2.2 循环语句 for … do

while 语句定义的循环结构是一种不定循环，即循环次数不是固定的。而 for … do 定义的是一种固定循环，其语法结构为：

```
for  变量名  in  值 1 … 值 n
do
     程序描述
done
```

在 for 结构中，变量依次匹配值列表中的各个值，每匹配一次则执行一次循环体，直至匹配完所有的值。

for 语句的语法结构图如图 19.4 所示。

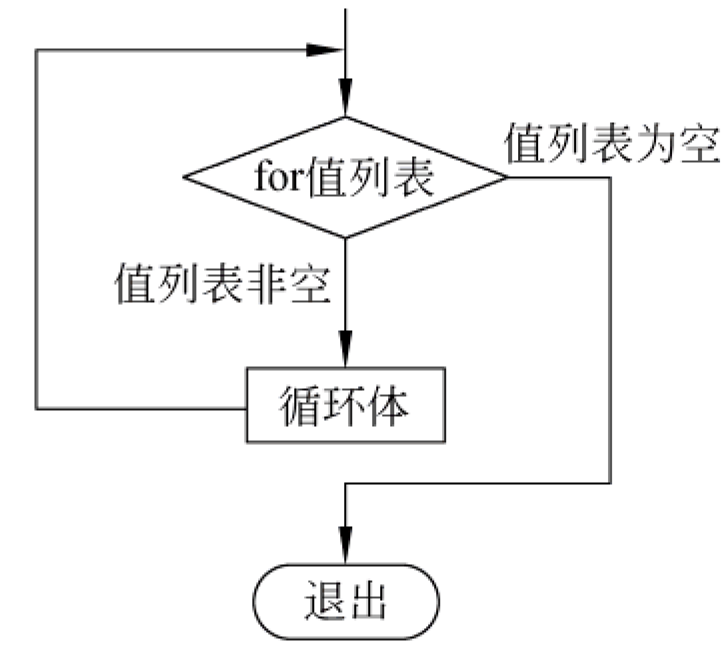


图 19.4　for 语句的语法结构图

【示例】显示系统中的用户名与登录 Shell。

```
#!/bin/bash

a='cut -d : -f 1,3 /etc/passwd | sort -t : -k 2 -n'
for name in $a
do
A='echo $name | cut -d : -f 1'
B='echo $name | cut -d : -f 2'
C='grep "$name" |cut -d : -f 7'
echo "$A UID is $B,login shell is $C"
done
```

19.2.3 控制语句的联合使用

本节利用上述介绍的控制语句来完成一个综合应用的脚本，该脚本用于探测网络中都有哪些 IP 地址被使用。

该脚本中将会使用 arping 这个命令，该命令用于向邻近主机发送 arp 请求，接收主

机会对 arp 请求进行回应，从而确认主机 IP 被占用。

下面对 arping 命令的用法加以说明。

【语法】**arping [选项] 目标 IP 地址**

选项说明如下：

-c *n*：*n* 为数字，用于定义发送 arp 请求包的数量。

-w *n*：*n* 为秒数，用于定义等待主机响应的时间。

```
#!/bin/bash
net=192.168.1.
for host in 'seq 1 100'
do
        arping -c 1 -w 1 $net$host &> /dev/null && ack=0 || ack=1
        if [ "$ack" == '0' ];
                then
                        echo "IP Address $net$host is onling"
                elif [ "$ack" == '1' ];
                        then
                                save="$save$net$host:"
        fi
done
echo $save > /tmp/test
```

在该脚本中联合使用了 if 判断语句和 for 固定循环语句。脚本中使用了 seq 命令，该命令用于输出一个数字序列。‘seq 1 100’表示输出数字从 1 开始到 100 结束。

```
[root@xinya script]# seq 1 10
1
2
3
4
5
6
7
8
9
10
```

最后，介绍 Shell Script 的追踪与调试命令。脚本的调试工作使用 sh 命令。

下面对 sh 命令的用法加以说明。

【语法】**sh [选项] Shell 脚本名**

选项说明如下：

-n：不执行 Shell 脚本，仅检查脚本语法问题。

-v：执行脚本前将脚本内容输出。

-x：将用到的脚本内容显示到屏幕上。

第 4 篇　网络服务基础

Linux 的主要应用方向之一就是网络功能，其强大的网络功能给了管理员一展拳脚的更多机会。但也正是这种强大、灵活的网络功能需要管理员投入更多的精力去维护。本篇内容就常见的 Linux 网络服务器的搭建与基本配置展开讨论，其目的是让读者对 Linux 网络服务器有一个基本的认识，为继续学习本书的高级版奠定良好的基础。

第20章　NFS网络文件系统

在由使用 Linux 操作系统的主机所组成的网络中，如何在网络上共享主机的文件资源是首先要解决的一个问题。这种需求在 Windows 操作系统网络中是通过文件共享来实现的，而在 Linux 操作系统网络中则是通过 NFS 服务来实现的。

NFS（Network File System，网络文件系统）是一种允许 UNIX/Linux 主机通过网络进行文件共享的网络协议。NFS 是由 SUN 公司开发的一种网络应用协议，主要用于在网络中提供目录资源的共享，该协议实际上是 RPC（远程过程调用）协议所管理的一种扩展功能。RPC 是一种支持进程间通过网络进行数据传输的协议，RPC 的 NFS 功能扩展使得 NFS 服务所共享的文件资源可通过网络进行传输。

NFS 服务采用客户端/服务器架构，如图 20.1 所示。

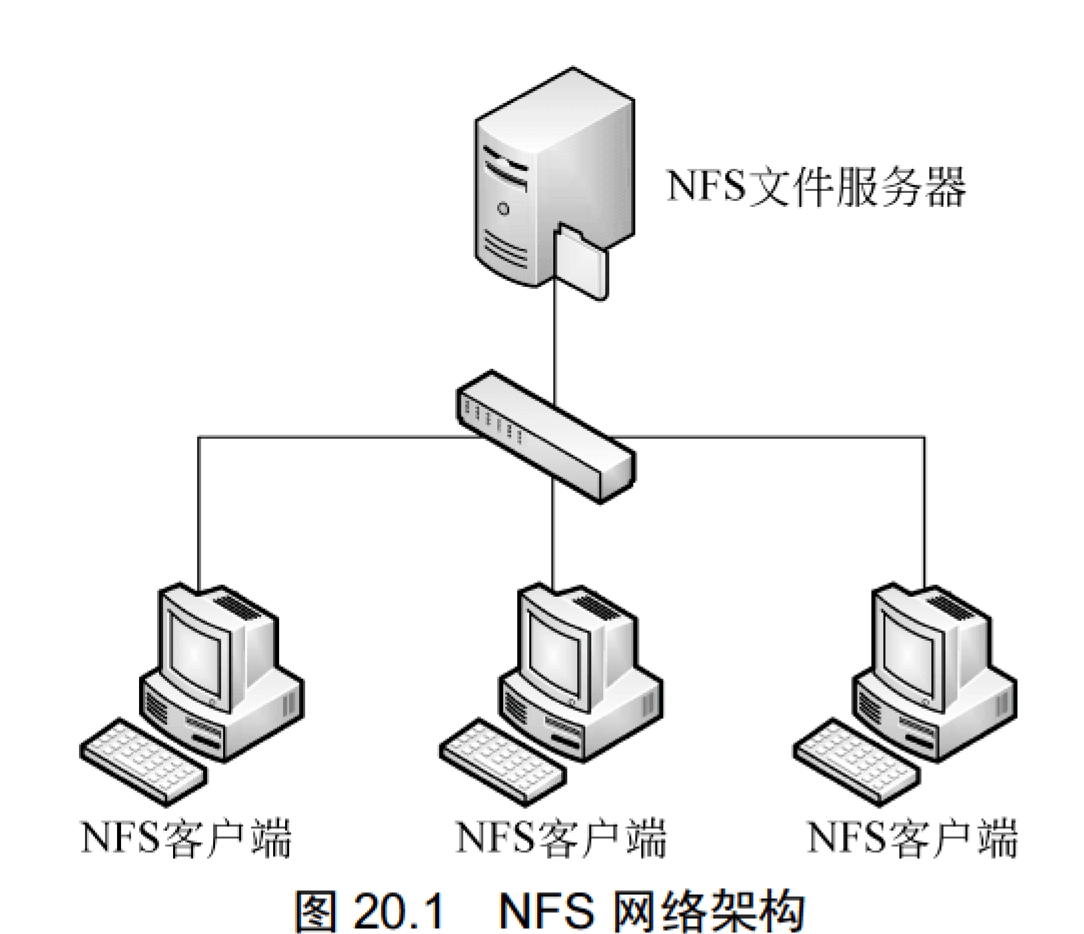


图 20.1　NFS 网络架构

（1）NFS 服务器的主要功能是将本地主机中的文件共享以接受 NFS 客户端的访问请求。

（2）NFS 客户端的主要功能是用于访问 NFS 服务器的共享资源。NFS 客户端连接至 NFS 服务器后，可以像使用本地文件系统一样使用 NFS 服务器的共享资源。

20.1　NFS 服务的安装

NFS 服务器与 NFS 客户端可以分别处于不同的主机中，也可以存在于同一主机中。事实上，大多数 Linux 主机都会以 NFS 服务器和 NFS 客户端的双重身份出现在网络中。

当前 Linux 发行版大多支持默认安装 NFS 服务器程序，CentOS 默认在安装操作系统时就已经安装好 NFS 服务器程序了。

例如，查询当前主机是否安装了 NFS 服务器程序：

```
[root@bogon ~]# rpm -q nfs-utils
nfs-utils-1.0.9-44.el5
```

```
[root@bogon ~]# rpm -q portmap
portmap-4.0-65.2.2.1
```

注意，前面已经提到过，NFS 服务是建立在 RPC 服务的基础上的，所以 NFS 服务在运行过程中需要 RPC 相关服务的支持，具体包括以下 3 项服务。

（1）portmap 服务：该服务为 RPC 服务提供端口映射功能，即当客户端要访问 RPC 服务所管理的具体服务时，portmap 服务会将具体服务所占用的端口返回给客户端，客户端依该端口进行访问。

（2）nfs 服务：该服务为基本 NFS 服务进程，主要用于管理客户端登录 NFS 服务器，判断哪些客户端有权访问本机 NFS 服务。

（3）rpc.mountd 服务：该服务是 RPC 服务的调用功能，用于管理客户端对共享资源的访问权限。在 NFS 客户端登录后，该服务会验证客户端对共享资源有哪些权限。

若当前系统并未安装 NFS 服务，则需要利用安装光盘中提供的 RPM 包进行安装。具体需要安装以下两个 rpm 包：

```
nfs-utils-1.0.9-44.el5.i386.rpm
portmap-4.0-65.2.2.1.i386.rpm
```

```
[root@bogon CentOS]# rpm -ivh nfs-utils-1.0.9-44.el5.i386.rpm
warning: nfs-utils-1.0.9-44.el5.i386.rpm: Header V3 DSA signature: NOKEY,
key ID e8562897
Preparing...          ########################################### [100%]
      package nfs-utils-1.0.9-44.el5.i386 is  installed
[root@bogon CentOS]# rpm -ivh portmap-4.0-65.2.2.1.i386.rpm
warning: portmap-4.0-65.2.2.1.i386.rpm: Header V3 DSA signature: NOKEY,
key ID e8562897
Preparing...          ########################################### [100%]
      package portmap-4.0-65.2.2.1.i386 is  installed
```

20.2 NFS 服务的控制

在完成 NFS 服务安装后，可启动 NFS 服务。NFS 服务的启动利用的是/etc/rc.d/init.d/nfs 脚本来进行控制的。在启动 NFS 服务的同时还需要启动 portmap 服务，以提供 NFS 服务的端口信息。portmap 服务的启动控制脚本为/etc/rc.d/init.d/portmap。

启动 NFS 服务的命令如下：

```
#/etc/rc.d/init.d/portmap start
#/etc/rc.d/init.d/nfs start
```

```
[root@bogon ~]# /etc/rc.d/init.d/portmap start
启动 portmap:                                              [确定]
[root@bogon ~]# /etc/rc.d/init.d/nfs start
启动 NFS 服务:                                             [确定]
关掉 NFS 配额:                                             [确定]
```

```
启动 NFS 守护进程:                                         [确定]
启动 NFS mountd:                                           [确定]
```

通过/etc/rc.d/init.d/nfs 服务控制脚本同样可以实现该服务的停止（stop）和状态查询（status）等操作。

在 NFS 服务运行过程中还可以利用 rpcinfo 命令来查看 NFS 以及相关的 RPC 服务的运行状态。

```
[root@bogon ~]# rpcinfo -p
  program vers proto   port
   100000  2   tcp    111  portmapper
   100000  2   udp    111  portmapper
   ⋮
   100003  2   udp   2049  nfs
   100003  3   udp   2049  nfs
   100003  4   udp   2049  nfs
   ⋮
   100005  3   udp    747  mountd
   100005  3   tcp    750  mountd
```

该命令会将与 NFS 服务相关的 portmap、NFS 与 mountd 服务所采用的传输协议以及相应的服务端口都显示出来。若在该命令的输出中不存在以上 3 个服务，则意味着 NFS 服务未正常启动。需要重新启动 NFS 服务。

20.3　NFS 服务的配置

NFS 服务的目的是要将本地文件系统中的某个目录共享到网络中供用户访问，实际上是要承担网络中的文件服务器的角色。但是究竟要将哪个目录共享出去，共享后网络用户对该目录有何种权限，以及网络中哪些用户可访问该共享资源，都需要具体配置。

对 NFS 服务的配置是通过 NFS 的主配置文件/etc/exports 来实现的，该文件的主要功能是定义 NFS 的输出目录（共享目录）、访问权限以及允许访问 NFS 服务的主机有哪些。

20.3.1　/etc/exports 文件的语法格式

/etc/exports 文件默认是个空文件，该文件的语法格式为：

【语法】<共享目录>　[客户端（权限，…，权限）]　[客户端（权限，…，权限）]

各选项说明如下：

共享目录：是指本机中待共享的目录。注意，NFS 是针对目录进行的共享，而不是针对文件进行的共享。当需要共享文件时，需要将文件存储至共享目录中实现。在文件系统中，建议为 NFS 服务定义一个专用的共享目录，如/share，被共享的文件可以通过符号链接或硬链接的方式存储于该目录中，这样有利于对共享资源的统一组织和管理。

客户端：是指允许访问本机 NFS 服务的计算。在/etc/exports 文件中，客户端计算机的表示方法有很多种，具体包括：

（1）使用 IP 地址表示单一主机，如 192.168.1.105。

（2）使用网络 IP 地址表示特定网络中的所有主机，如 192.168.1.0/24 或 192.168.1.*。

（3）使用完全合格的域名表示单一主机，如 abc.shenghao.com。

（4）使用泛域名表示特定域中的所有主机，如*.shenghao.com。

（5）使用“*”表示任何主机。

权限 ：是指客户端访问该服务器共享资源时所具有的权限与相关操作，具体可分为 3 个方面。

（1）访问权限，包括：

ro：客户端以只读方式访问共享资源。

rw：客户端以可读写方式访问共享资源。

（2）用户映射操作，即客户端将以何种身份访问 NFS 服务器的共享资源，包括：

all_squash：将所有客户端的用户与组默认映射为 nfsnobody，即客户端是以 nfsnobody 身份访问共享资源。

no_all_squash：不进行 nfsnobody 映射，即客户端用户以其登录身份访问共享资源，这是 NFS 服务的默认配置。

root_squash：将 root 用户映射为 nfsnobody 用户，这是 NFS 的默认配置。即客户端以 root 用户身份访问 NFS 的共享资源时，其将获得 nfsnobody 用户的身份与权限。

no_root_squash：不将 root 用户映射为 nfsnobody 用户。

anonuid=UID：将所有客户端用户身份映射为特定的用户。

anongid=GID：将所有客户端用户所属组映射为特定的组。

（3）链接安全选项，包括：

secure：限制客户端只能使用小于 1024 的端口访问 NFS 服务器，这是 NFS 服务器的默认配置。

insecure：客户端可以使用大于 1024 的端口访问 NFS 服务器。

sync：数据同步写入，这样做的效率比较低，但是有利于数据的一致性。

async：数据异步写入，数据会先保存在内存的缓冲区中，在适当的时候会被写入硬盘。

wdelay：延期执行，即将相关操作一并执行，以便提高执行效率，这是 NFS 的默认配置。

no_wdelay：若有操作则立即执行，该选项应与 sync 一并使用。

subtree_check：共享目录时，NFS 服务器将检查其父目录的权限，这是 NFS 服务器的默认配置。

no_subtree_check：共享目录时，NFS 服务器不检查其父目录的权限，这有助于提高访问效率。

以上是/etc/exports 文件的语法结构，这里需要注意的是共享权限的问题。如果在共享资源时不定义任何共享权限，则 NFS 将采用默认权限。这时共享资源的权限将成为：

```
(no_all_squash, root_squash, secure, wdelay, subtree_check)
```

在这默认权限定义中，未包括用户是否对共享资源可读写以及是否执行数据同步写入的定义。所以，在配置共享资源时，至少要定义 ro 或 rw 以及 sync 或 async。

20.3.2　NFS 共享的配置示例

为演示实际的共享操作，首先定义一个专供 NFS 服务使用的共享目录/share。

```
[root@bogon /]# mkdir share
[root@bogon /]# ls -ld share
drwxr-xr-x 2 root root 4096 Jul 23 05:46 share
```

【示例】共享目录/share/public 允许 192.168.1.0 网络中的所有主机以可读可写的方式访问，对于其他网络中的用户则只能以可读的方式访问。

```
/share/public  192.168.1.*( rw, sync)  *( ro )
```

这里需要注意的一点是，本例中尽管 NFS 支持 192.168.1.0 网络中的所有用户可对/share/public 目录进行读取与写入操作，但是否能够真的完成该操作还要看本地文件系统的定义。

若本地文件系统未开放访问用户对目录的写入操作，则 NFS 客户端仍旧无法实现对共享资源的写入操作。

本例中， /share/public 目录的本地权限为 755。

```
drwxr-xr-x 2 root root 4096 Jul 23 05:50 public
```

所以尽管 NFS 允许对该目录进行写入操作，但由于本地权限未开放，所有 NFS 客户端还是无法执行写入操作。

```
[user1@hero mnt]$ touch file1
touch: cannot touch 'file1': Permission denied
```

在这种情况下，只有开放本地权限中的其他用户可写权限，NFS 客户端才能正常地访问完整的共享资源。

```
[root@bogon /]# chmod o+w /share/public
[root@bogon /]# ll /share
drwxr-xrwx 2 root root 4096 Jul 23 05:50 public
```

客户端执行写入操作

```
[user1@hero mnt]$ touch file1
[root@hero mnt]# ll
total 0
-rw-r--r-- 1 nfsnobody nfsnobody 0 Jul 23 05:57 file1
```

【示例】共享/share/user 目录，仅供 192.168.1.20 客户端以可读写方式访问。

```
/share/user  192.168.1.20(rw,sync)
```

【示例】共享/share/file 目录，供 192.168.2.0 网络中的所有主机以只读的方式访问，并且将所有访问用户均映射为 nfsnobody 用户。

```
/share/file  192.168.2.0/24(ro,all_squash,sync)
```

20.3.3 NFS 服务的共享列表

当完成 NFS 服务的共享定义后，可以通过更新 NFS 服务的共享列表使共享定义即刻生效。NFS 服务共享列表的维护命令为 exportfs。

exportfs 命令的作用是更新和显示 NFS 服务的共享信息，并且可以暂停某 NFS 目录的共享。

下面对该命令的用法加以说明。

【语法】**exportfs** [选项]

选项说明如下：

-a：显示/etc/exports 文件所定义的所有共享目录。

-r：重新读取/etc/exports 文件的共享定义信息，使共享定义立即生效。

-u：停止共享某一目录。

-v：将执行结果显示在屏幕上。

查看当前 NFS 服务共享了哪些目录资源：

```
[root@bogon etc]# exportfs -av
exporting 192.168.1.20:/share/user
exporting 192.168.2.0/24:/share/file
```

重新读取/etc/exports 文件，使修改立即生效：

```
[root@bogon etc]# exportfs -rv
exporting 192.168.1.20:/share/user
exporting 192.168.2.0/24:/share/file
exporting 192.168.1.*:/share/public
exporting *:/share/public
```

20.3.4 NFS 服务的维护

在 NFS 运行过程中，需要适时地监控该服务的运行状态。这种监控主要包括两个方面：

（1）对共享目录的权限进行监控。

（2）对共享目录的使用情况进行监控。

尽管在/etc/exports 文件中定义了共享目录的权限，但实际上共享目录所采用的默认权限还有很多。

对共享目录的默认权限，可以通过查看/var/lib/nfs/etab 文件得知，该文件记录了共

享目录的配置信息。

```
[root@bogon ~]# cat /var/lib/nfs/etab
/share/user
192.168.1.20(rw,sync,wdelay,hide,nocrossmnt,secure,root_squash,no_all_
squash,no_subtree_check,secure_locks,acl,mapping=identity,anonuid=65534,
anongid=65534)
/share/file
192.168.2.0/24(ro,async,wdelay,hide,nocrossmnt,secure,root_squash,all_
squash,no_subtree_check,secure_locks,acl,mapping=identity,anonuid=65534,
anongid=65534)
/share/public
192.168.1.*(rw,sync,wdelay,hide,nocrossmnt,secure,root_squash,no_all_
squash,no_subtree_check,secure_locks,acl,mapping=identity,anonuid=65534,
anongid=65534)
/share/public
*(ro,sync,wdelay,hide,nocrossmnt,secure,root_squash,no_all_squash,no_
subtree_check,secure_locks,acl,mapping=identity,anonuid=65534,anongid=
65534)
```

在 NFS 管理过程中除对共享目录的权限进行监控外，还涉及对共享目录的使用情况进行监控。

对共享目录的使用情况进行监控时可以使用 showmount 命令。

【语法】**showmount** [选项]　**NFS 服务器 IP 地址|服务器名**

选项说明如下：

当不指定服务器 IP 地址或服务器名时，默认显示的是本机 NFS 服务器的信息。

-a：显示当前与 NFS 服务器链接的所有客户端及其链接目录。

-d：显示 NFS 服务器中所有已被客户端链接的共享目录。

-e：显示 NFS 服务器上所有的共享目录。

20.4　NFS 客户端的访问

NFS 服务器配置完成后，NFS 客户端可以直接访问。对于 NFS 客户端而言，可以使用“showmount -e”命令查看 NFS 服务器上都有哪些共享资源，然后利用 mount 命令将 NFS 服务器上的共享资源挂载到本地文件系统中，实现直接访问。

查看 NFS 服务器上的共享资源的命令格式如下：

showmount　-e　NFS 服务器 IP 地址

```
[root@hero ~]# showmount -e 192.168.1.105
Export list for 192.168.1.105:
/share/public (everyone)
/share/file   192.168.2.0/24
/share/user   192.168.1.20
```

挂载 NFS 服务器上特定的共享资源的命令格式如下：

mount -t nfs NFS 服务器 IP 地址：共享目录名 本地文件系统挂载点

```
[root@hero~]# mount -t nfs 192.168.1.105:/share/file  /mnt/nfs
```

注意，当客户端无权访问 NFS 共享资源时，这种挂接操作将不能完成，并提示访问被禁止。

```
[root@hero ~]# mount -t nfs 192.168.1.105:/share/user  /mnt/nfs
mount: 192.168.1.105:/share/user failed, reason given by server: Permission
denied
```

卸载 NFS 网络文件系统与卸载普通挂载资源类似，都是使用 umount 命令来完成，格式如下：

umount 挂载点

```
[root@hero ~]# umount /mnt/nfs
```

对于 NFS 共享资源而言，如果需要客户端系统每次启动系统后自动挂载，就需要编辑/etc/fstab 文件来实现 NFS 共享资源的启动自动挂载。

在/etc/fstab 文件中需要添加如下内容：

NFS 服务器 IP 地址:NFS 共享目录 本地文件系统挂载点 nfs defaults 0 0

如每次启动时均自动挂载 192.168.1.105 服务器上的/share/public 共享目录，在其/etc/fstab 文件中的写法为：

```
192.168.1.105: /share/public  /mnt/nfs  nfs  defaults 0  0
[root@bogon ~]# vi /etc/fstab
LABEL=/                  /            ext3    defaults         1 1
tmpfs                    /dev/shm     tmpfs   defaults         0 0
devpts                   /dev/pts     devpts  gid=5,mode=620   0 0
sysfs                    /sys         sysfs   defaults         0 0
proc                     /proc        proc    defaults         0 0
LABEL=SWAP-sda2          swap         swap    defaults         0 0
192.168.1.105:/share/public /mnt/nfs  nfs     defaults         0 0
```

本章就 NFS 进行了讨论，NFS 不仅简化了 Linux 网络环境下的文件共享问题，而且提高了存储资源的利用率，为客户端计算机节省了磁盘空间。由于可将 NFS 服务器作为网络服务的中间节点，这就给文件的集中管理创造了基本条件。

通过对 NFS 的学习会发现，NFS 仅可以在 Linux 操作系统之间实现资源的共享，若需要在不同操作系统之间（如 Windows 与 Linux 操作系统之间）进行资源共享，NFS 就无能为力了。解决异构操作系统之间资源的共享问题的最好方法是利用 Samba 服务，有关 Samba 服务的内容将在下一章展开讨论。

第21章　Samba 服务的配置与应用

实际工作环境中的网络通常为混杂网络，所谓混杂网络是指网络中包含有多种操作系统，如 Linux、UNIX 和 Windows 等。上一章解决了 Linux 操作系统之间文件资源共享的问题。那么在不同操作系统之间要如何实现文件资源甚至是打印设备等资源的共享呢？Samba 服务能很好地解决这一问题。它能够使 Windows 用户通过“网上邻居”等熟悉的方式直接访问 Linux 上的共享资源，也能使 Linux 用户利用 Samba 客户端程序访问 Windows 的共享资源，解决了混杂网络中的资源共享这一问题。

21.1　Samba 概述

Samba 是一个应用程序的名称，这个应用程序实际上是利用 SMB(Server Message Block，服务信息块)网络协议实现在不同操作系统间共享资源的操作。

SMB 协议可以实现在局域网上共享文件与打印机，该项协议可以用在多种应用层协议之上，通过 SMB 协议，客户端应用程序可以在各种网络环境下读、写服务器上的文件，以及对服务器提出服务请求。

此外通过 SMB 协议，应用程序还可以访问远程服务器端的文件和打印机等资源。现在 SMB 协议可以应用在包括 Linux 在内的很多平台上。

作为 SMB 协议在 Linux 操作系统上的实现程序，Samba 可以在现有的 Linux 系统上提供 SMB 服务，使 Windows 用户能够访问并使用 Linux 操作系统共享的文件和打印机。同样，Linux 用户也可以通过 SMB 服务来使用 Windows 上的共享文件和打印机资源，因为 Windows 本身就提供了 SMB 协议，在 Windows 中实现 SMB 服务的是 Server 服务。

Samba 是 GPL 授权软件，用户可以合法且免费地使用该软件。Samba 软件的官方网站为 http://www.samba.org，有关 Samba 的维护、支持与更新信息均可通过该站点获得。

CentOS 5 中所提供的 Samba 的版本为 3.0，也是 Samba 的最新版本。该版本的 Samba 的主要功能为：

（1）提供文件和打印机共享，支持 Windows 客户端访问。

（2）支持 NetBIOS 名称解析，可以为 Linux 和 Windows 操作系统提供名称解析服务，并可承担 Windows 网络中的主控浏览器角色和 WINS 服务器角色。

（3）提供 SMB 客户功能。利用 Samba 提供的 smbclient 程序可以在 Linux 上以类似于 FTP 的方式访问 Windows 的资源。

（4）提供一个命令行工具。利用该工具可以有限制地支持 Windows 的某些管理功能。

21.2 Samba 服务的安装

在 CentOS 5 中，Samba 软件包是默认安装在系统上的，可以使用“rpm -q samba”命令查询系统当中是否已经安装了 Samba 软件包。

```
[root@dns CentOS]# rpm -q samba
samba-3.0.33-3.28.el5
```

如果系统中没有安装这个软件包，则可将光盘放入光驱，将光驱挂载到系统中，到挂载目录下的 CentOS 目录中去查找并安装这个软件包。

```
[root@dns CentOS]# rpm -ivh samba-3.0.33-3.28.el5.i386.rpm
warning: samba-3.0.33-3.28.el5.i386.rpm: Header V3 DSA signature: NOKEY,
key ID e8562897
Preparing...          ########################################### [100%]
 1:samba              ########################################### [100%]
```

21.3 Samba 服务器的配置

Samba 服务的配置是通过配置文件来完成的，Samba 服务的主配置文件为/etc/samba/smb.conf。Samba 服务的大部分功能都在这个文件中设置，文件中有许多不同的配置选项。Samba 的设置比较繁杂，在这里主要介绍一些常用的配置。

21.3.1 /etc/samba/smb.conf 文件的格式

/etc/samba/smb.conf 文件由两部分构成。

（1）全局设置（Global Settings）：用于定义与 Samba 服务整体运行环境有关的配置，它的设置项目是针对所有共享资源的。

（2）共享定义（Share Definitions）：用于定义共享目录的设置，只对被共享的资源起作用。

在配置文件中，以分号和#号作为注释符。如果该行以这些符号开头，则该行的内容会被忽略而不会生效。配置文件的格式是以“设置项目=设置值”的方式来表示的。

21.3.2 Samba 服务的用户身份验证

与 Samba 服务用户身份验证相关的文件有两个。

1. /etc/samba/smbpasswd

该文件用于保存 Samba 用户的用户名与密码，Samba 服务在安装完成后，这个文件是不存在的。

在一些发行版中可以使用 smbpasswd 命令来建立这个文件，当用户第一次使用 smbpasswd 命令为 Samba 服务添加用户账号时，会自动建立 smbpasswd 文件：

```
smbpasswd  -a  linux 账号名
```

但在比较新的发行版本当中，由于使用了 tdbsam 用户验证机制，系统默认会使用 tdbsam 的安全账户管理模式。如果要使用 smbpasswd 这种认证方式的话，需要编辑配置文件，将 passdb backend =tdbsam 机制关闭（# passdb backend =tdbsam），然后加入“smb passwd file = /etec/samba/smbpasswd”启用用户验证文件，存储到 smbpasswd 文件。这样可以使用 smbpasswd 这种认证方式进行身份认证了。

smbpasswd 文件默认是空的，里面没有任何用户信息，还需要用前面提到的 smbpasswd 命令来向文件中添加用户。

Samba 服务与 Linux 系统使用不同的密码文件，因此无法使用 Linux 系统中的账号登录 Samba 服务器。也就是说，一个可以登录 Samba 服务器的 Samba 用户必须先成为 Linux 系统当中的用户。但是，如果系统当中已经存在一个用户，但并没有将该用户加入到 smbpasswd 文件中，那么这个用户是无法登录到 Samba 服务器当中去的。

2. /etc/samba/smbusers

该文件是用于控制用户映射，即将访问 Samba 服务器的用户映射为 Samba 用户。

21.3.3 Samba 服务的日志文件

Samba 服务的日志默认存放在/var/log/samba 目录中，Samba 服务为所有连接到 Samba 服务器的计算机分别建立日志文件，同时也将 NMB 服务和 SMB 服务的运行日志分别写入 smbd.log 和 smbd.log 日志文件中。

21.4 Samba 服务的基本配置

21.4.1 全局参数

全局参数用于设置 Samba 服务的整体运行环境，它的设置参数很多，但通常在实际应用中并不需要全部进行设置。主要的设置参数如下。

（1）设置 Samba 服务器所属的群组名称或 Windows 的域名。

可以使用 workgroup=MYGROUP 项目进行设置。可将此名称设置为与被提供服务的 Windows 操作系统中的名称相同，以方便用户的访问。例如，被提供服务的 Windows 操作系统中的群组名为 xinya，则此项目设置为：

```
Workgroup = xinya
```

（2）设置 Samba 服务器的简要说明。

可使用 server string=Samba Server 项目进行设置。将它修改为有关服务器的简要说

明，方便访问用户的识别。例如：

```
Server string = xinya's  Samba Server
```

（3）设置可访问 Samba 服务器的主机、子网或域。

可使用 hosts allow=192.168.1. 192.168.2. 127.项目进行设置。如果设置的项目超过一个，就必须用空格或逗号隔开。默认此配置是不使用的，即所有的主机都可以访问，所以要使用时需要将行首的分号删除。

如当前要配置服务允许主机名为 sales 的客户端访问，允许域名为 xinya.com 的域访问，允许 192.168.0.*的网络中除 192.168.0.20 主机外的所有主机均可访问，则设置如下：

```
hosts allow = sales，xinya.com，192.168.0. EXCEPT  192.168.0.20
```

（4）设置 Samba 服务启动时，将自动加载打印机配置文件。

可使用 printcap name=/etc/printcap 项目进行设置。建议使用默认值。

（5）设置是否允许打印配置文件中的所有打印机开机时自动加载。

可使用 load printers=yes 进行设置，使 Samba 服务启动时自动加载浏览列表。默认为 yes，即允许自动加载浏览列表，可使用默认值。

（6）设置 guest 账号名。

可使用 guest account=pcguest 项目进行设置。在此设置的账号名都必须新建到 /etc/passwd 文件中。如果未指定，服务器就会以 nobody 账号来处理。默认的配置是不使用，可使用默认值。

（7）指定 Samba 服务器使用的安全等级。

可使用 security=user 项目进行设置，默认值为 user。Samba 服务器的安全等级共有 5 类。

① share：当客户端连接到该级别的 Samba 服务器时，不需要输入账号和密码，就可以访问 Samba 服务器上的共享资源，但安全性无法得到保障。

② user：在客户端连接到该级别的 Samba 服务器时，访问该服务器的共享资源前，用户需要输入有效的账号和密码，通过验证后才能使用服务器的共享资源。默认的配置为该等级，但最好使用加密的方式传送密码，以提高安全性。

③ server：与 user 级别等级相同，也需要输入有效的账号和密码，但密码的验证会由另一台 SMB 服务器负责，因此还必须指定口令服务器，即设置 password server 选项。

如果验证失败，服务器就会使用 user 安全等级进行访问。需要注意的是，如果采用加密的密码，Samba 服务器就无法反向检查原有的密码文件，所以必须指定另一个有效的 smbpasswd 密码文件。

④ domian：Samba 服务器加入到 Windows NT 域中后，Samba 服务器不再负责账号和密码的验证，统一交由域控制器负责，使用该安全等级，同时也必须指定口令服务器。

⑤ ads：Samba 服务器加入到 Windows 活动目录后，使用该安全级别。同时也需指定口令服务器。

（8）有多个网卡的 Samba 服务器设置需要监听的网卡。

可通过 interfaces=网卡 IP 地址或网络接口设置该功能。在默认的配置下并不使用，但为了保证多网卡的 Samba 服务器能正常工作，应设置此项。

比如当前配置的 Samba 服务器中有两块网卡，分别为 eth0 和 eth1，它们所对应的 IP 地址分别为 192.168.0.20 和 202.19.23.233。如果设置监听的网卡为 eth0，可以做如下设置：

```
interfaces = eth0
```

或

```
interfaces = 192.168.0.20
interfaces = 192.168.0.20/24
interfaces = 192.168.0.20/255.255.255.0
```

（9）设置 Samba 服务器同时充当 WINS 服务器。

利用 wins support=yes 来设置是否将这台 Samba 服务器作为 WINS 服务器，默认是不使用。如果想使用，只需要将配置文件中该语句前的分号注释去掉即可。

（10）设置 WINS 服务器的 IP 地址。

一台 Samba 服务器不能同时作为 WINS 服务器和客户端。如果一台 Samba 服务器不是 WINS 服务器，但又需要 WINS 服务，可设置“wins server=w.x.y.z”项目来指定 WINS 服务器，同时这台 WINS 服务器还必须能在 DNS 服务器中登记。设置如下：

```
Wins server = 192.168.0.20
```

21.4.2 用户映射

用户映射在 Windows 和 Linux 主机之间进行。两个系统拥有不同的用户账号，用户映射的目的就是将不同的用户映射成为一个用户。做了映射后的 Windows 账号在使用 Samba 服务器上的共享资源时就可以直接使用 Windows 账号进行访问。

全局参数“username map”就是用来控制用户映射的，它允许管理员指定一个映射文件，该文件包含了在客户机和服务器之间进行用户映射的信息。默认情况下指定的映射文件为/etc/samba/smbusers。

要使用用户映射，只需要将 smb.conf 配置文件中 username map=/etc/samba/smbusers 前的分号注释去掉，然后编辑文件/etc/samba/smbusers，将需要映射的用户添加到文件中。格式为：

```
单独的 Linux 账号=要映射的 Windows 账号列表
```

账号列表内当有多个用户时，使用空格隔开。如要在 Windows 下的用户 alice 和 robin 和 Linux 中的用户 tom 之间建立映射，可做如下设置：

```
tom=alice  robin
```

21.4.3　使用加密码口令

目前多数操作系统都已使用加密的方式来发送密码，因为采用这种方式能防止类似嗅探器一类的工具轻易地获取用户密码。

全局参数“encrypt password”项目可设置将指定用户的密码是否以加密的方式发送到 Samba 服务器，默认值是使用此功能。

参数格式如下：

```
encrypt password = yes / no
```

使用 yes 表示采用加密的方式发送密码，使用 no 则不采用加密方式。

Windows 操作系统也采用加密的方式发送密码，如果此参数设置为 no，就必须修改 Windows 系统的注册表。为了简化用户的操作，Samba 提供了多种 Windows 操作系统类型的注册表文件，这些文件存放在/usr/share/doc/samba*/registry 目录下，用户可以根据 Windows 操作系统的类型选择相应的文件，将它复制到 Windows 客户端后直接运行。

21.4.4　共享目录

1. 设置用户主目录

Samba 服务为每一个 Samba 用户提供一个主目录，该共享目录通常只有用户本身可以使用。用户主目录默认存放在/home 目录下，每个 Linux 用户有一个独立的子目录。相关设置如下：

```
[homes]
comment = Home Directories
browseable = no
writable = yes
```

（1）共享说明

将“comment=目录描述”设置为简要的主目录说明描述，方便用户使用。

（2）是否允许用户浏览所有人的主目录

browseable 设置是否允许用户浏览/homes 目录。默认值为不允许用户浏览其他用户的主目录。

（3）是否允许写入个人主目录

writable 项目设置是否允许用户写入自己的主目录，no 为不允许用户写入自己的主目录，yes 是允许用户写入自己的主目录。

该段设置的作用范围与全局参数类似，对所有的用户起作用，无法为个别用户单独设置。

2. 设置一个自定义共享目录

除了用户的主目录外，通常还需要根据实际设置其他的共享目录，为用户提供服务。如图 21.1 所示。

在图 21.1 中[share]为共享目录名，也就是该目录被共享后访问者看到的名称，该名称和共享目录名可不相同。

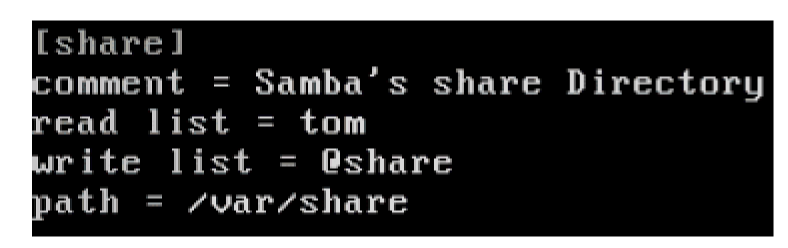

[share]
comment = Samba's share Directory
read list = tom
write list = @share
path = /var/share

图 21.1　设置自定义共享目录

read list 指定了有哪些用户可以读取该共享目录中的内容，图 21.1 中指定了 tom。

write list 指定了有哪些用户可以向该共享目录中写入，图 21.1 中在 share 前加了一个@符号，此符号的作用是告诉系统，这代表的不是用户，而是一个组，以@符号将用户与组区分开。

如果所有用户的读写权限是相同的，就可以使用 writable=yes/no 来设置，yes 允许所有有效账号读写，否则使用 no。

path 指定了共享目录的本地绝对路径。此路径后指定的共享目录必须真实存在，否则用户在访问时会提示访问错误。

21.5　Samba 服务的打印共享

为了节约打印资源，公司中很少会为每一台计算机都配置打印机，最好的办法就是将打印机共享，使需要的用户通过网络访问并使用它们。默认情况下，Samba 的打印服务是开放的。当管理员在服务器上安装设置好打印机后，其他用户就可以使用了。

与共享目录不同，打印机安装完成后，Samba 服务必须重新启动，否则客户端可能无法看到该项打印机。

首先将打印机安装到系统中，如果 Windows 用户要使用 Samba 提供的共享打印服务，还必须安装该打印机相应的 Windows 版本的驱动程序。

与共享打印有关的配置主要在配置文件中的[printers]部分，内容如下：

```
[ printers ]
comment = All Printers
path =  /var/spool/samba
browseable = no
guest OK = no
writable = no
printable = yes
```

上面的配置与共享目录是基本相同的，如果允许 guest 打印，只需在末尾加入 public=yes 即可。

21.6　Samba 服务的启动和停止

21.6.1　启动 Samba 服务

启动 Samba 服务：

```
[root@dns CentOS]# service smb start
启动 SMB 服务:                                             [确定]
启动 NMB 服务:                                             [确定]
```

21.6.2 停止 Samba 服务

停止 Samba 服务：

```
[root@dns CentOS]# service smb stop
关闭 SMB 服务:                                             [确定]
关闭 NMB 服务:                                             [确定]
```

21.6.3 重新启动 Samba 服务

重新启动 Samba 服务：

```
[root@dns CentOS]# service smb restart
关闭 SMB 服务:                                             [确定]
关闭 NMB 服务:                                             [确定]
启动 SMB 服务:                                             [确定]
启动 NMB 服务:                                             [确定]
```

21.7 Linux 客户端的访问

在 Linux 主机上，可以利用 smbclient 程序来连接 Windows 或 Samba 服务器上的共享资源，smbclient 采用类似于 FTP 命令行的环境，功能强大，使用方便。

smbclient 是 Linux 操作系统默认安装在系统上的，可以使用“rpm -q samba-client”命令检查系统当中是否安装了 smbclient。

```
[root@dns CentOS]# rpm -q samba-client
samba-client-3.0.33-3.28.el5
```

如果系统当中没有安装该软件包，则需要用户手工安装。

安装完成后，就可以使用它来访问 Samba 服务器了，使用格式如下：

smbclient -L 主机名或 IP 地址 -U 登录用户名

```
[root@dns CentOS]# smbclient -L //192.168.1.101 -U user1
Password:
Domain=[XINYA] OS=[Unix] Server=[Samba 3.0.33-3.28]

       Sharename      Type     Comment
       ---------      ----     -------
       share          Disk     Samba's share Directory
       are            Disk     Samba's share Directory
```

```
        IPC$            IPC      IPC Server (Samba Server Version 3.0.33-3.28 )
        User1           Disk     Home  Directories
Domain=[XINYA] OS=[Unix] Server=[Samba 3.0.33-3.28]

        Server                   Comment
        ---------                -------
        Workgroup                Master
        ---------                -------
        MSHOME                   vivi7E
        MYGROUP                  XINYA
        WORKGROUP                WWW
```

在示例中，上面是所属的组、操作系统类型和版本，然后是所连接主机目前的共享资源，下面是有关主机的描述和群组名称等信息。

如果要连接 192.168.1.110 主机上的 share 共享文件夹，则在链接时直接声明被访问共享目录名：

```
[root@dns CentOS]# smbclient -L //192.168.1.101 -U user1
Password:
Domain=[XINYA] OS=[Unix] Server=[Samba 3.0.33-3.28]
smb: \> ls
```

smbclient 的使用方式和 FTP 客户端相似，可以使用 get 和 put 命令下载和上传文件，同时也可以使用 help 命令获得帮助。

21.8　Windows 客户端的访问

Windows 的客户端不需要更改任何设置，在“网上邻居”中的 Mygroup 工作组中就可以查看到安装了 Samba 的 Linux 服务器，或者在开始菜单中的“运行”中输入“\\服务器名称”或“\\服务器 IP 地址”，单击“确定”按钮即可，如图 21.2 所示。



图 21.2　Windows 客户端的访问

第22章 DNS服务器的基本配置

在计算机网络中是使用 IP 地址来标识主机的网络接口的，但是 IP 地址具有含义不明确、不便于记忆的特点。为此网络中往往采用将 IP 地址与便于识别和记忆的名称联系起来，通过使用这些名称来找到 IP 地址，进而访问到目标主机。可以将这种将 IP 地址与名称联系起来的过程称为“IP 地址与名称的映射”。

与主机 IP 地址映射的名称有两种，一种是主机名，另一种是域名。

利用主机名与 IP 地址映射在网络中标识主机的方法是计算机网络中最早使用的，且目前仍在使用的一种方法。这种映射方法是当前局域网中的主机广泛使用的方法，但这种方法要求同一局域网中的主机名具有唯一性，且主机名无法与所在网络、所提供的服务等信息联系起来，所以主机名在当前主要应用于内部局域网中。

在 Linux 操作系统中可使用 hostname 命令来查看和定义当前主机的主机名，Linux 操作系统中对主机名的定义是在/etc/sysconfig/network 文件中使用 HOSTNAME 命令来定义的。

继主机名之后，当前网络中大量使用了“域名与 IP 地址映射”这种方法来标识目标计算机，特别是在 Internet 上，域名已经成为了事实上的网络接口名称标准。域名具有含义清楚、便于记忆、能明确反映主机所在网络和所提供的服务的特点，与主机名相比，域名的应用范围更加广泛，甚至在 Internet 上也依靠域名来定位目标主机的位置。

无论是采用主机名还是采用域名来与 IP 地址进行映射，都涉及如何查找名称所对应的 IP 地址的问题，可以将查找名称所映射的 IP 地址的过程称为“名称解析”。

22.1 名称解析方法概述

在网络实践中，名称解析的方法主要有 3 种：

（1）利用 Host 本地数据库进行名称解析。

（2）利用 NIS 进行名称解析。

（3）利用 DNS 进行名称解析。

22.1.1 利用 Host 本地数据库进行名称解析

利用 Host 本地数据库进行名称解析是最早使用的名称解析方法，开始仅支持对网络中的主机名进行解析，现在也开始支持对域名的解析。

Host 解析机制是在主机的本地文件系统中创建 Host 数据库文件，Linux 操作系统中这个 Host 数据库文件是/etc/hosts。该文件中包含了网络中的主机名与 IP 地址映射关系的记录列表，而该列表是管理员手工创建的。

/etc/hosts 文件的语法结构为：

```
IP 地址   主机名   域名
```

这里以本机默认的/etc/hosts 文件为例来说明该语法结构。文件中以#号开头的为注释信息，同时文件中的每一行就是一条映射记录。

```
[root@dns named]# cat /etc/hosts
# Do not remove the following line, or various programs
# that require network functionality will fail.
127.0.0.1       localhost.localdomain       localhost
IP 地址          域名                         主机名
::1             localhost6.localdomain6     localhost6
IPV6 地址        域名                         主机名
```

当网络中出现新的主机时，需要手工将主机名信息添加至该文件中。例如网络中新增加了两台主机，其中一台的主机名为 dns，其 IP 地址为 192.168.1.105；另一台主机的域名为 www.xinya.com，其 IP 地址为 192.168.1.108。管理员需要手工将映射信息添加至/etc/hosts 文件中。

```
# Do not remove the following line, or various programs
# that require network functionality will fail.
127.0.0.1           localhost.localdomain localhost
::1                 localhost6.localdomain6 localhost6
192.168.1.105       dns
192.168.1.108       www.xinya.com
```

当网络中的计算机需要访问网络中的名为 dns 的主机时，会首先查找/etc/hosts 文件中是否存在 dns 主机对应的 IP 地址的映射记录，如果有，则直接使用该映射记录中的 IP 地址访问目标主机，在本例中会直接使用 192.168.1.105 这个 IP 地址访问目标主机。

Host 本地数据库是专用分散管理方式，当网络中一台主机的映射信息发生变化时，网络中所有的主机都要在本机的/etc/hosts 文件中反映该变化。特别是在使用 DHCP 动态分配 IP 地址的网络中，hosts 文件常常会因为主机 IP 地址的变化而不得不被频繁修改。

当前/etc/hosts 文件在网络中主要充当备份资源，特别是在 DNS 离线的情况下，/etc/host 文件可以暂时负责名称解析工作。

22.1.2 利用 NIS 进行名称解析

与 Host 名称解析机制相比，NIS 名称解析机制有了很大的进步。

NIS 这种解析机制采用客户机/服务器方式来实现名称解析。网络中的主机作为 NIS 服务器的客户端而存在，而 NIS 服务器中会维护一个网络中所有主机与 IP 地址的映射表。

当 NIS 客户端以主机名方式发起链接访问时，NIS 客户端会首先向 NIS 服务器查询被访问主机名对应的 IP 地址，NIS 服务器收到查询请求后，会依服务器内存储的映射信息对客户端进行响应。

NIS 服务器保存着网络中所有主机的 IP 与主机名的映射信息，使/etc/hosts 文件的分散名称解析机制转化为集中管理与解析机制。这种机制克服了 Hosts 机制下工作量大、数据同步性差的缺点。

但是，由于 NIS 服务器仅支持在 Intranet 网络中提供名称解析服务，对于更大的网络，NIS 服务不能更好地支持。因此，NIS 服务往往应用于企业内部网络，随着当前 DNS 名称和解析机制的广泛使用，NIS 服务逐渐被 DNS 所替代。

22.1.3　利用 DNS 进行名称解析

DNS 名称解析是当前最为流行的一种名称解析方式，支持对域名系统的完整解析。由于在 Internet 上采用域名标识目标主机，因而作为专注于域名解析的 DNS 服务已成为当前网络架构的基础服务。

DNS 名称解析机制采用客户端/服务器这种工作模式，网络中的主机作为 DNS 服务器的客户端，当需要解析域名对应的 IP 地址时会主动向 DNS 发起询问。DNS 在接收到客户端的请求后，会根据本地数据库的记录查询该域名所对应的 IP 地址，并将查询结果返回给用户。

DNS 这种工作方式与 NIS 服务的工作方式类似，但与 NIS 服务不同的是。当 DNS 不能解析客户端提交的域名所对应的 IP 地址时，DNS 会查询网络中的其他 DNS 服务器。DNS 服务器会沿着域名所表达的网络信息持续查找下去，直到得知域名所对应 IP 的信息。这种查找方式使得 DNS 的作用范围超出了本地网络的限制，形成了一种可以延伸至整个 Internet 的一种名称解析机制。

22.2　DNS 服务的基本要素

前已述及，DNS 是一种采用客户端/服务器方式进行域名解析的服务系统。在 DNS 系统中涉及域名空间、DNS 服务器与 DNS 客户端 3 个基本要素。

本节首先分析 DNS 基本要素的结构，而后结合 DNS 的基本要素说明 DNS 名称解析机制的工作原理。

22.2.1　域名空间

DNS 服务器是用于进行域名解析的，也就是将域名对应的 IP 地址查找出来并发送给 DNS 客户端。那么，什么是域名？域名是如何构成的？这是我们首先要讨论的一个问题。

所谓域名是一种在网络中标识主机的名称，以克服使用 IP 地址标识主机所带来的识别与记忆上的困难。

与主机名不同的是，域名具有结构性特点，即依据域名可以得知域名所标识的主机在哪个网络中，这一点是主机名所无法比拟的。这里以 www.xinya.com 这个域名为例来说明域名的结构性特点。

www.xinya.com 这个域名的完整写法为 www.xinya.com，注意简写域名与完整域名之间唯一的差别就在于最后一个“.”。这个“.”表示的是根域。而 www.xinya.com.这个域名表示的这台主机在根域下的.com 这个网络下的.xinya 这个网络中，是.xinya 网络中名为 www 的主机。

从以上叙述可见，域名结构为“主机名.二级域名.顶级域名 根域”。那么，“.”根域在哪里？.com 网络在哪里?.xinya 这个网络又在哪里呢？这就涉及域名空间的问题了。

域名并不是允许用户任意指定的，当然如果在内部网络中，出于试验与学习的需要是可以任意定义的，但是在 Internet 中域名的取得是需要申请后付费使用的。

域名系统是由总部位于美国的 InterNIC 来定义和维护的，InterNIC 将域名系统划分成若干个层次，如图 22.1 所示。

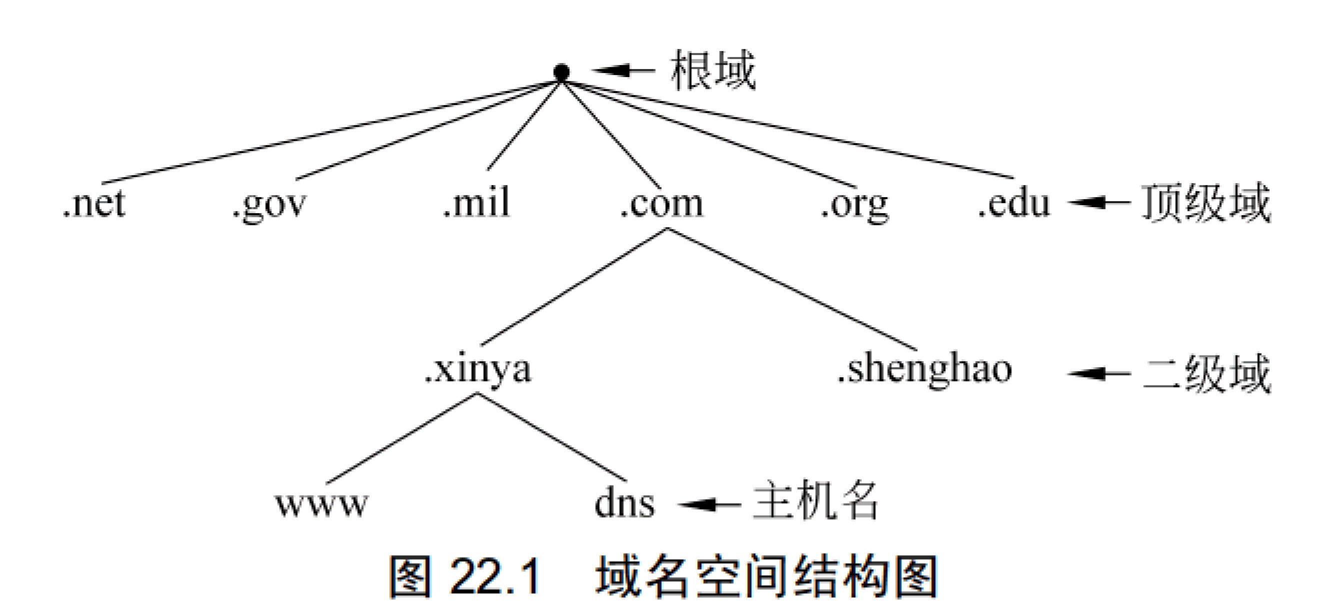


图 22.1　域名空间结构图

InterNIC 定义，Internet 中的所有域名都应位于域名空间最上层的根域的管理之下，而“.”根域是由 InterNIC 来维护和管理的。也就是说，Internet 中的所有域名都可以沿着根域向下一步一步查找，最终查找到主机名所对应的 IP 地址。

在 DNS 结构中，当 DNS 客户端向 DNS 服务器递交了域名的解析请求时，如果本地 DNS 服务器不知道该域名对应的 IP 地址，就会向根域发出查询请求，沿着根域的指引，一步一步查询到域名所标识的主机的 IP 地址。

而根域的 DNS 服务器地址是什么呢？InterNIC 定义的根域一共向外提供 13 个 DNS 服务器，供来自世界各地的 DNS 提交查询请求。这里需要注意的是，根域中并未保存所有的域名映射信息，根域的 DNS 中仅仅保存着其下顶级域的 DNS 服务器地址。

顶级域是 InterNIC 定义的，位于“.”根域之下的，数量有限且不能轻易变动的域名。顶级域存在的目的是对网络中庞大的域名进行分类。InterNIC 定义的顶级域包括很多种，但总体上可分为两类。

（1）机构域：按机构性质所进行的分类，常见的机构域域名包括：

.com：表示商业机构。

.edu：表示教育机构。

.net：表示网络技术机构。

.mil：表示军事机构。

.gov：表示政府机构。

.org：表示非营利性组织。

（2）地理域：按机构所处的位置进行的分类，常见的地理域域名包括：

.CN：中国。

.HK：中国香港特别行政区。

.TW：中国台湾省。

.US：美国。

.UK：英国。

.JP：日本。

对于一个机构而言，在向 InterNIC 提交域名申请时必须指明要将自己的域名放置到哪个顶级域之下。

每个顶级域都配备有自己的 DNS 服务器，该 DNS 服务器的地址要在“.”根域中备案，以便根域能随时掌握顶级域中 DNS 服务器的地址信息。

在顶级域下就是二级域，二级域是由机构自己定义的并向 InterNIC 申请获得的。InterNIC 规定，在相同的顶级域下，二级域的域名具有唯一性，也就是说在.com 这个顶级域下不能存在两个名称为 xinya 的二级域。

在机构向 InterNIC 申请域名时，除提供要申请的域名外，还需要提供负责解析该域名下的主机或子域 IP 地址的 DNS 服务器 IP 地址，InterNIC 会将该 DNS 的 IP 地址备份到顶级域的 DNS 服务器中。

在二级域下还可以继续存在三级域、四级域等子域，这取决于使用机构的实际需要。但是不论有多少个子域，在域名的最左端都是主机名。

例如 www.xinya.com.这个域名表示的是根域下的.com 顶级域下的.xinya 二级域下的 www 这台主机。而 mail.mark.xinya.com.这个域名表示根域下的.com 顶级域下的.xinya 二级域下的.mark 三级域下的 mail 这台主机。

在域名结构中，最左侧的是主机名。主机名理论上是可以任意定义的，但是在网络服务构建时常常会有一些约定俗成的应用习惯。如网络中提供 Web 服务的主机名为 www，提供邮件服务的主机名为 mail，提供 FTP 服务的主机名为 ftp，提供新闻组服务的主机名为 news，等等。这也是为什么 Internet 上的 Web 站点经常使用 www 作为主机名的原因，这实际上是一种习惯用法。

在表述一个域名时，如 www.xinya.com，这种方式实际上是这个域名的简写方式，其完整的写法为 www.xinya.com.，注意不要忘了最后的根域。这种带有根域表示的完整域名称为“完全合格域名”，简称 FQDN。

22.2.2 DNS 服务器与客户端

DNS 服务器是 DNS 名称解析系统中提供名称解析的服务程序，该服务器会保存其所负责解析的域中的主机名与 IP 地址的映射信息。在域名空间中，每一级域名空间都会配置有相应的 DNS 来负责解析该域下的主机或子域的 IP 地址信息。

DNS 客户端是 DNS 名称解析系统中负责提交域名解析请求并接收域名解析结果的应用程序名，该应用程序安装在网络中的主机上。当网络中的主机产生域名请求时，该客户端程序就会首先向 DNS 服务器发出解析域名的请求，DNS 服务器完成名称解析后会将域名对应的 IP 地址返回给 DNS 客户端。

22.2.3 DNS 名称解析的过程

在了解了 DNS 服务的基本要素后，下面以这些基本要素为基础来讨论 DNS 名称解析的过程。DNS 采用的是“客户机/服务器”这种工作模式，在 DNS 查询过程中存在 DNS 客户机向 DNS 服务器的查询与 DNS 服务器之间的查询。如图 22.2 所示的查询是一种简单的查询，即 DNS 客户端向 DNS 服务器直接查询其所负责解析的区域数据。

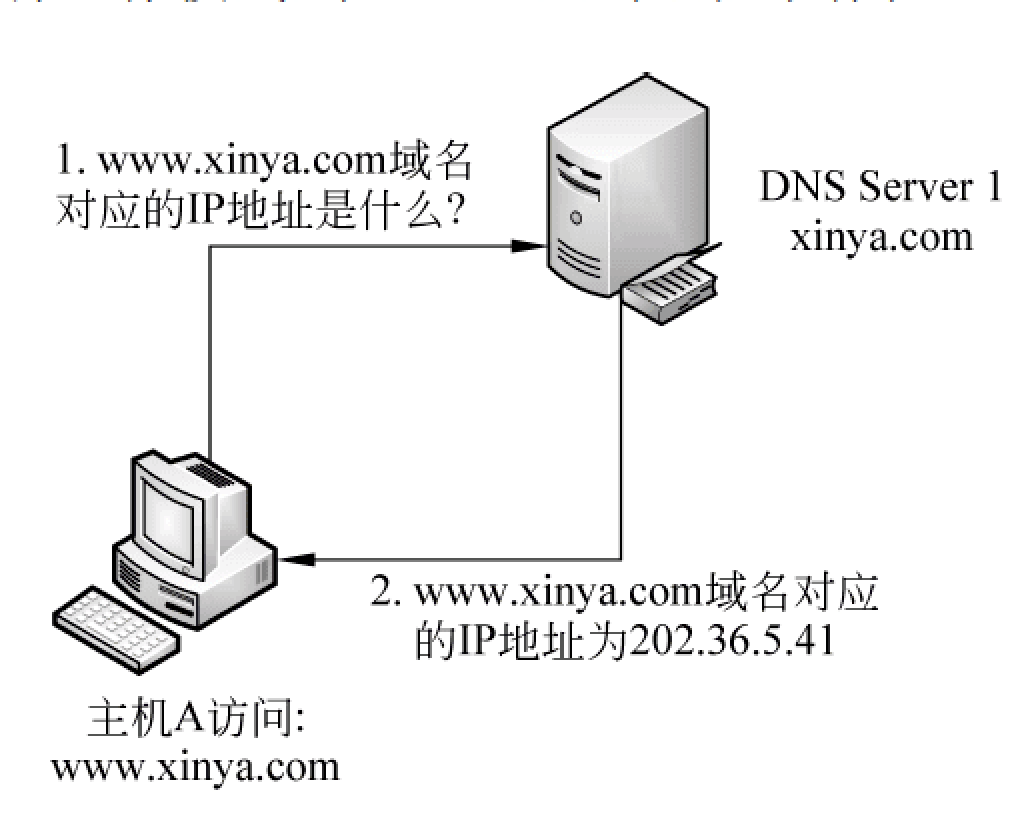


图 22.2 本区域的 DNS 查询过程

在图 22.2 中，作为 DNS 客户端的主机 A 需要访问 www.xinya.com 这个域名，该访问可以是主机 A 中的应用程序发起的，如主机 A 的网络浏览器的 URL 地址为 http://www.xinya.com，或者主机 A 使用了“ping www.xinya.com”这条命令，这都将导致主机 A 需要获得 www.xinya.com 这个域名所对应的 IP 地址。

此时，主机 A 的 DNS 客户端程序会将“www.xinya.com 域名对应的 IP 地址是什么？”这个查询提交给它的 DNS 服务器要求解析，该 DNS 服务器在图 22.2 中为 DNS Server1。DNS Server 1 接到查询请求后会根据域名来判断其能否解析该域名。本例中，客户端要求解析的是 www.xinya.com 这个域名，而该 DNS 正好是负责 xinya.com 这个域的名称解析工作，因此 DNS Server 1 会直接查询本地数据库，以确定 www.xinya.com 域名所对应的 IP 地址。若 DNS Server 1 的本地数据库中存在 www.xinya.com 域名所对应的 IP 地址的映射信息，DNS Server 1 会将该 IP 地址响应给客户端（主机 A）。若 DNS Server 1 的本地数据库中不存在 www.xinya.com 域名所对应的 IP 地址的映射信息，则 DNS Server 1 会向客户端返回 www.xinya.com 域名不存在的错误消息。

在 DNS 查询过程中，除上述查询过程外，还会出现另一种查询问题，即当主机 A 向 DNS 提出查询请求，而 DNS 并不负责该域名的解析工作，则 DNS 将如何解析域名呢？见图 22.3 所示。

在图 22.3 所示的 DNS 查询过程中，主机 A 需要查询 www.shenghao.com 这个域名所对应的 IP 地址。此时主机 A 将该查询请求提交给 DNS Server 1，DNS Server 1 在接收到该域名后，发现该域名应该由 shenghao.com 区域的 DNS 服务器进行解析，而自己只负责解析 xinya.com 域的主机信息，此时 DNS Server 1 要找到 shenghao.com 这个域的 DNS 服务器，并要求 shenghao.com 的 DNS 服务器解析 www.shenghao.com 这个域名。但 DNS

Server 1 将如何找到 shenghao.com 的 DNS 服务器呢？其查找 shenghao.com 的 DNS 服务器的过程如下。

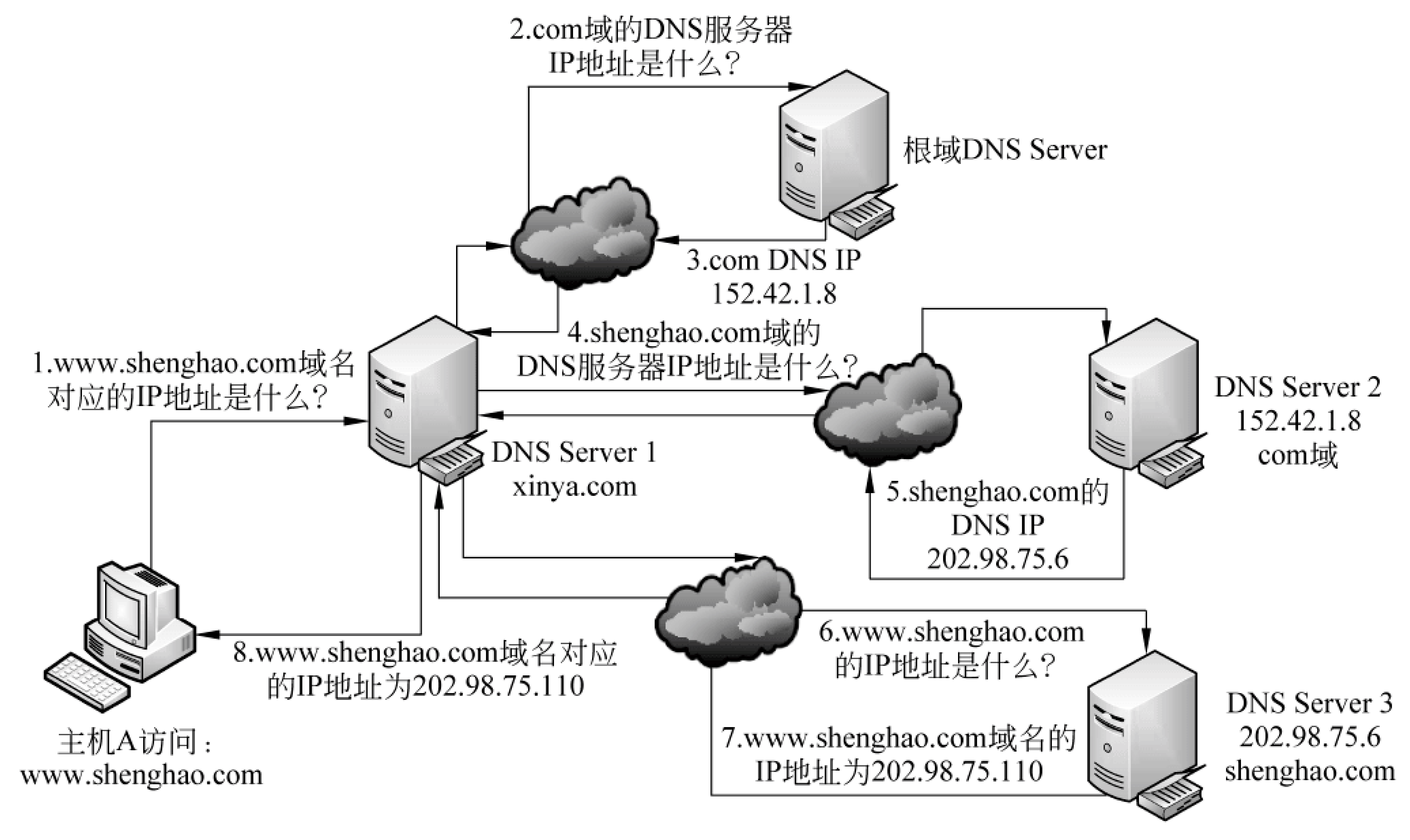


图 22.3　DNS 查询过程

（1）首先，DNS Server 1 会向根域发起查询，因为从域名结构上看，www.shenghao.com 是注册在.com 域下的一个域名，在 shenghao 向.com 提交注册申请时必然同时提交了负责解析该域信息的 DNS 服务器地址，因此要找到 shenghao.com 域的 DNS 服务器就首先要找到.com 这个域的 DNS，向其查询 shenghao 域的 DNS 信息。但是，.com 域的 DNS 服务地址为何 DNS Server1 也不清楚，但 DNS Server 1 知道根域的 DNS 服务器的 IP 地址（根域 DNS 服务器地址包含在 DNS 服务器的 named.ca 文件中），为此 DNS Server 1 向根域发起了“解析.com 域”的查询请求，根域 DNS Server 在查询本地数据库后对该请求进行了回应，通知 DNS Server 1：.com 域 DNS 服务器的 IP 地址为 152.42.1.8。

（2）在获知.com 域 DNS 服务器的 IP 地址为 152.42.1.8 后，DNS Server 1 会向.com 域的 DNS 服务器 152.42.1.8 地址发起“解析 shenghao.com 域”的查询请求，.com 域的 DNS Server 2 在查询本地数据库后对该请求进行了回应，通知 DNS Server 1：shenghao.com 域 DNS 服务器的 IP 地址为 202.98.75.6。

（3）在获知 shenghao.com 域 DNS 服务器的 IP 地址为 202.98.75.6 后，DNS Server 1 会向 shenghao.com 域的 DNS 服务器 202.98.75.6 地址发起“解析 www.shenghao.com 主机”的查询请求，shenghao.com 域的 DNS Server 3 在查询本地数据库后对该请求进行了回应，通知 DNS Server 1：www.shenghao.com 域 DNS 服务器的 IP 地址为 202.98.75.110。

（4）DNS Server 1 在获得 www.shenghao.com↔202.98.75.110 的映射信息后，会首先将该信息存储到本地缓存中，以便客户端再次查询该域名时直接调用。同时会将该映射关系发回给客户端主机 A。

至此，DNS Server 1 完成了对 www.shenghao.com 这个域名的解析过程。通过该过程

可见，客户端与 DNS 之间实际上进行的是一次交互过程，即客户端提出查询请求，而服务器响应客户端的查询请求。但是 DNS 服务器为了正确响应客户端的查询请求，实际上执行了若干次与其他 DNS 服务器的交互式查询，包括与根域 DNS 服务器之间的交互、与.com 域 DNS 服务器之间的交互，以及与 shenghao.com 域的 DNS 服务器之间的交互，如图 22.4 所示。

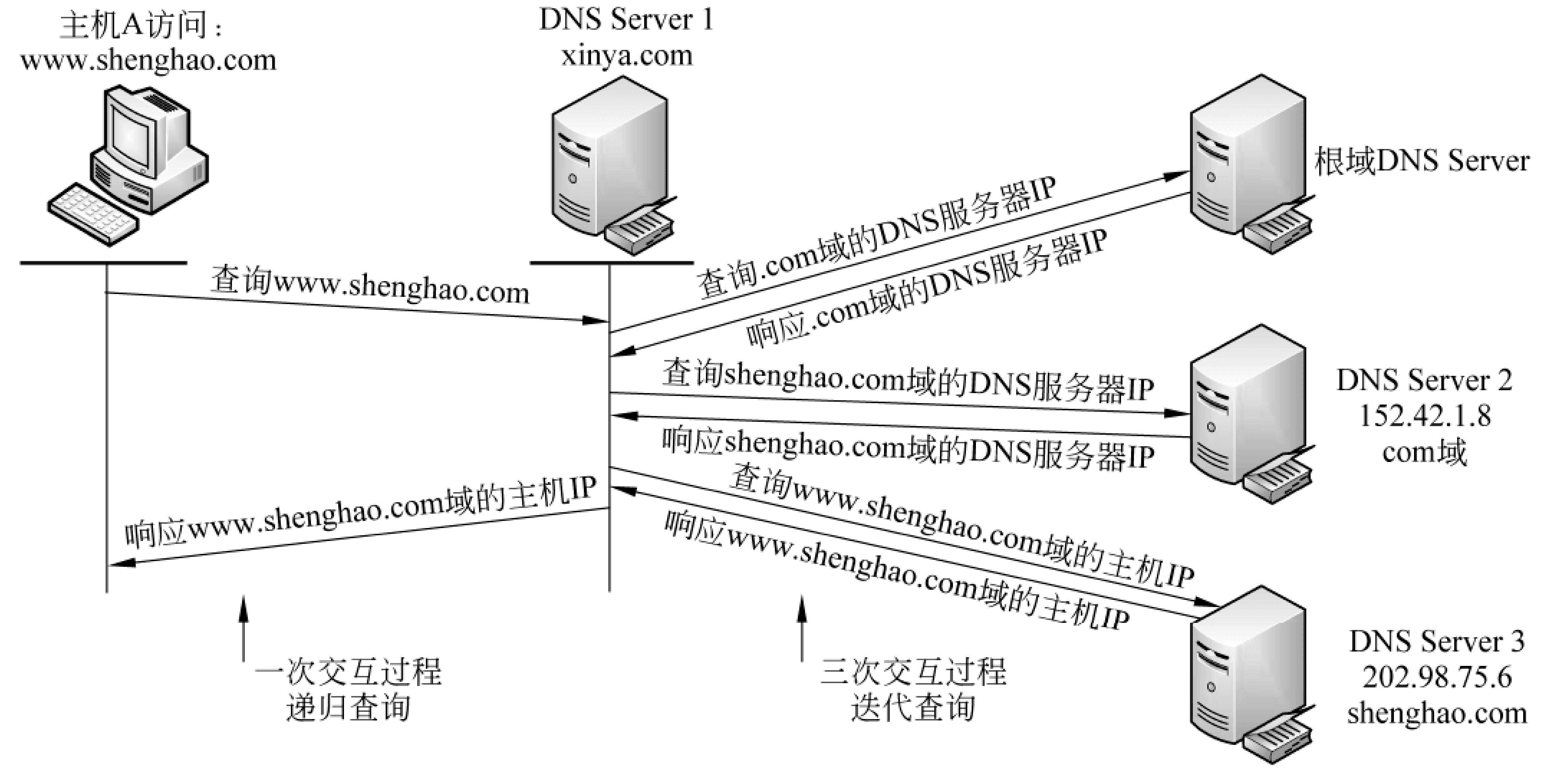


图 22.4　DNS 服务器的查询过程

由图 22.4 可见，在整个 DNS 查询过程中共分为两种形式的查询。

（1）递归查询：DNS 客户端与 DNS 服务器之间所进行的查询，整个查询过程只是一次交互。

（2）迭代查询：DNS 服务器之间的查询，整个查询过程与被查询的域名结构有关。DNS 服务器通过这种迭代查询实现了对客户端的透明化查询。对于 DNS 客户端而言，不需要考虑 DNS 迭代查询的问题，客户端将解析请求提交后静等查询结果即可。

通过以上的介绍，现在可以对 DNS 服务下一个定义了，所谓 DNS 服务是以域名空间结构为基础，利用递归查询和迭代查询将域名解析为 IP 地址的服务。

22.2.4　DNS 服务器的种类

一台 DNS 服务器既可以负责一个域中“主机↔IP”映射信息的解析工作，也可以负责多个域中“主机↔IP”映射信息的解析工作。在 DNS 服务器中，将其负责解析的每个域称为一个“区域”，每个“区域”中的“主机↔IP”映射信息数据称为“区域数据”。当一台 DNS 服务器负责多个域的解析工作时，该 DNS 服务器将为每个被解析的域建立一个“区域”，并将被解析的域的映射信息存储到该“区域”的“区域数据文件”中。

对于 DNS 服务器而言，对区域的解析分为两种。

（1）正向区域解析：即将域名解析为 IP 地址。对于正向区域解析，DNS 服务器要事先建立“正向解析区域”与“正向区域数据文件”，当正向解析请求发送过来之后，查

找正向区域数据文件即可实现解析。

（2）反向区域解析：即将 IP 地址解析为域名。反向解析需求在网络中主要是供一些应用程序使用，如防火墙的反向解析需求、邮件系统的反垃圾邮件需求等。对于反向区域解析，DNS 服务器要事先建立“反向解析区域”和“反向区域数据文件”，当反向解析请求发送过来之后，查找反向区域数据文件即可实现解析。

DNS 服务器作为网络应用的基础服务器，其重要性是不言而喻的。为保证 DNS 服务器能完成 7×24 的全时在线服务，常常为 DNS 服务器提供备份主机，以在 DNS 服务器宕机或离线的情况下继续提供名称解析服务。同时，为了节约维护成本，在一些不具备部署 DNS 服务器条件的网络环境中也提供本地 DNS 服务，常常借助于 DNS 的缓存服务器来实现对本地 DNS 服务器的访问。

基于以上需求，网络中的 DNS 服务器常常被划分为 3 种类型，如图 22.5 所示。

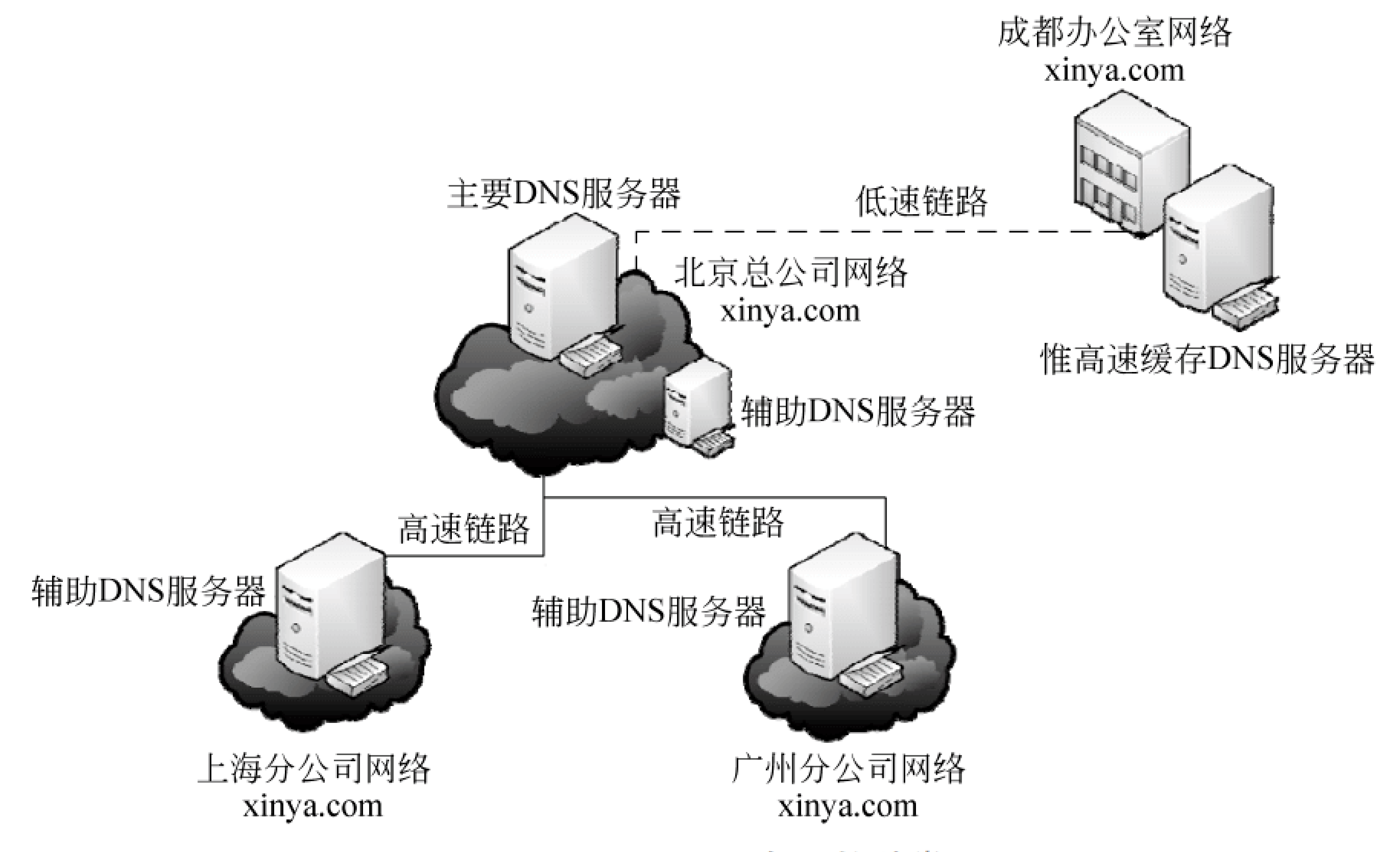


图 22.5　DNS 服务器的种类

（1）主要 DNS 服务器：保存着其所解析区域的源数据正本，是区域数据添加、修改和变更的位置。主要 DNS 服务器会向辅助 DNS 服务器提供区域数据，供辅助 DNS 服务器备份和解析使用。

（2）辅助 DNS 服务器：保存着其所解析区域的数据副本，其数据由主要 DNS 服务器复制而来。辅助 DNS 服务器的作用表现在两个方面。

① 作为主要 DNS 服务器的备份服务器，当主要 DNS 服务器离线时可继续提供域名解析。

② 向外提供域名解析，这实际上分担了主要 DNS 服务器的流量负担。

（3）惟高速缓存服务器：这种 DNS 服务器的类型比较特殊，该服务器中并未保存区域数据信息，仅记录了负责区域解析的 DNS 服务器的 IP 地址。惟高速缓存服务器也会接受 DNS 客户端的解析请求，但它不能解析域名，而是将名称解析请求发送给其所记录的 DNS 服务器解析。DNS 服务器将解析结果返回给惟高速缓存服务器后，惟高速缓存

服务器会在将解析结果缓存下来的同时将结果响应给客户端计算机。惟高速缓存服务器常应用于分支机构网络环境，以提高内部 DNS 查询的效率。同时也需要注意，DNS 服务器的迭代查询操作实际上也是借助于惟高速缓存服务器机制来实现的，所以说一般 DNS 服务器都具有惟高速缓存服务器的功能。

22.3　BIND 服务的安装与启动

在 Linux 操作系统中，实现 DNS 服务的是 BIND 程序，BIND（Berkeley Internet Name Domain，伯克利网络域名服务）是由美国加州大学伯克利分校开发和维护的，用于提供 DNS 服务功能的应用程序，CentOS 5 中所采用的 BIND 应用程序的版本为 9.3。

下面以一个典型应用模型为例介绍 BIND 的配置过程，模型如图 22.6 所示。

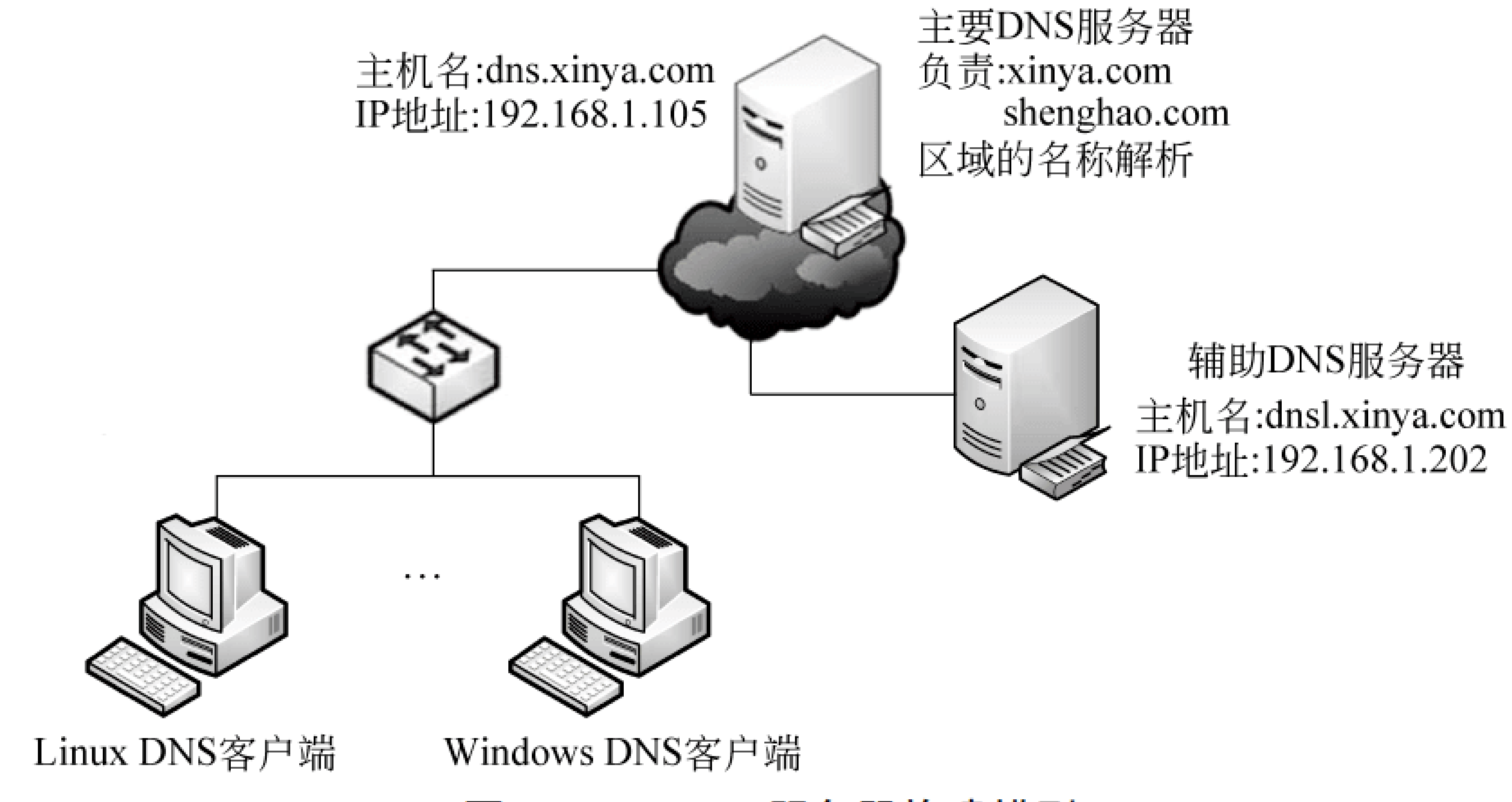


图 22.6　BIND 服务器构建模型

DNS 服务器使用的是客户端/服务器（C/S）网络架构，在该架构中主要 DNS 服务器主机为 dns.xinya.com，其 IP 地址为 192.168.1.105，向网络中的客户端计算机提供 xinya.com 与 shenghao.com 两个域名的名称解析，同时利用迭代查询机制为客户端提供其他域名的名称解析工作。

图 22.6 中的 dns1.xinya.com 是辅助 DNS 服务器，一方面可以作为主要 DNS 服务器的备份使用，另一方面也可以为网络中的 DNS 客户端计算机提供名称解析。

图 22.6 所示的 DNS 服务网络的构建顺序分为主要 DNS 服务器的搭建与配置、客户端的搭建与配置以及辅助 DNS 服务器的搭建与配置 3 部分。

（1）主要 DNS 服务器的搭建与配置包括以下任务。

① BIND 服务器的安装与控制。

② 定义 BIND 服务的区域。

③ 定义 BIND 服务的区域数据文件。

（2）客户端的搭建与配置包括以下任务。

① Linux 客户端的配置。

② Windows 客户端的配置。

③ DNS 客户端工具。

（3）辅助 DNS 服务器的搭建与配置。

22.3.1 主要 DNS 服务器的安装与启动

在配置 DNS 服务前，首先要在本机安装 BIND 服务器程序，BIND 服务器程序的 RPM 格式安装包位于 CentOS 安装光盘的 CentOS 目录下。

```
[root@dns /]# mount /dev/cdrom  /media/centos
mount: block device /dev/cdrom is write-protected, mounting read-only
[root@dns /]# cd /media/centos/CentOS/
[root@dns CentOS]# ls bind*
bind-9.3.6-4.P1.el5_4.2.i386.rpm
bind-chroot-9.3.6-4.P1.el5_4.2.i386.rpm
bind-devel-9.3.6-4.P1.el5_4.2.i386.rpm
bind-libbind-devel-9.3.6-4.P1.el5_4.2.i386.rpm
bind-libs-9.3.6-4.P1.el5_4.2.i386.rpm
bind-sdb-9.3.6-4.P1.el5_4.2.i386.rpm
bind-utils-9.3.6-4.P1.el5_4.2.i386.rpm
```

在安装 BIND 服务程序时，需要 3 个软件包：

（1）bind-9.3.6-4.P1.el5_4.2.i386.rpm：BIND 主程序包。

（2）bind-utils-9.3.6-4.P1.el5_4.2.i386.rpm：BIND 查询工具软件（默认已安装）。

（3）caching-nameserver-9.3.6-4.P1.el5_4.2.i386.rpm：惟高速缓存 DNS 服务器的配置文件。

可以使用“rpm -q a | grep bind”命令来查看当前系统是否安装了 BIND 服务程序，若未安装可以使用“rpm -ivh”命令来进行安装。

```
[root@dns CentOS]# rpm -q bind  ←查询是否安装了 BIND 服务程序
package bind is not installed
[root@dns CentOS]# rpm -q bind-utils  ←查询是否安装了 bind-utils
bind-utils-9.3.6-4.P1.el5_4.2  ←作为 DNS 客户端工具，系统默认已安装 bind-utils
[root@dns CentOS]# rpm -q caching-nameserver
                                        ←查询是否安装了 caching-nameserver
package caching-nameserver is not installed
[root@dns CentOS]# rpm -ivh bind-9.3.6-4.P1.el5_4.2.i386.rpm  ←安装 BIND
warning: bind-9.3.6-4.P1.el5_4.2.i386.rpm: Header V3 DSA signature: NOKEY,
key I D e8562897
Preparing...          ########################################### [100%]
   1:bind             ########################################### [100%]
[root@dns  CentOS]#  rpm  -ivh  caching-nameserver-9.3.6-4.P1.el5_4.2.
i386.rpm
warning: caching-nameserver-9.3.6-4.P1.el5_4.2.i386.rpm: Header V3 DSA
signature: NOKEY, key ID e8562897
```

```
Preparing...          ########################################### [100%]
   1:caching-nameserver  ########################################### [100%]
```

安装完成后，BIND 服务的守护进程服务名为 named，其服务控制脚本为/etc/rc.d/init.d/named，可以使用该脚本来控制 BIND 服务的启动与关闭，在 CentOS 系统中也可以使用"service named start|stop|restart"命令来启动、关闭或重新启动 named 服务。需要注意的是，每次对 BIND 服务进行修改后都需要重新启动 BIND 服务以便使修改生效。

```
[root@dns ~]# /etc/rc.d/init.d/named start      ←启动 named 服务
启动 named:                                        [确定]
[root@dns ~]# /etc/rc.d/init.d/named stop       ←停止 named 服务
停止 named:                                        [确定]
[root@dns ~]# /etc/rc.d/init.d/named restart    ←重新启动 named 服务
停止 named:                                        [确定]
启动 named:                                        [确定]
```

22.3.2 定义 BIND 服务所解析的区域

BIND 服务在安装完成并启动后会读取其主配置文件，以确定其所负责解析的区域、区域数据文件的存储位置以及 BIND 服务在运行过程中的相关参数。所以对 BIND 服务的调整和配置是需要通过修改配置文件来实现的。

BIND 服务的主配置文件为/etc/named.conf，在安装完 BIND 服务后，该文件默认为空文件或者该文件根本就不存在。对于这个主配置文件，BIND 服务提供了一个示例文件，该示例文件为/usr/share/doc/bind-9.3.6/sample/etc/named.conf，可以以该示例文件为模板来定义/etc/named.conf 文件，若/etc/named.conf 文件并不存在则可以直接创建该文件。

```
[root@dns ~]# cat /usr/share/doc/bind-9.3.6/sample/etc/named.conf
//
// Sample named.conf BIND DNS server 'named' configuration file
// for the Red Hat BIND distribution.
…
options
{
      …
       directory "/var/named"; // the default
       dump-file              "data/cache_dump.db";
       statistics-file        "data/named_stats.txt";
       memstatistics-file     "data/named_mem_stats.txt";

};
…
view "internal"
{
```

```
…
     zone "my.slave.internal.zone" {
         type slave;
         file "slaves/my.slave.internal.zone.db";
         masters { /* put master nameserver IPs here */ 127.0.0.1; } ;
         // put slave zones in the slaves/ directory so named can update them
     };
```

由示例文件可见，文件中每一条定义语句都以“;”为结尾。文件中以#或//开头的行为注释行，行中使用/*与*/符号括起来的内容也为注释内容。

/etc/named.conf 文件从总体上可以分为两个部分：全局配置部分与区域配置部分。

（1）全局配置部分：用于定义 BIND 服务器的工作属性，涉及整个服务器共享的运行信息。全局配置部分使用 options 命令来定义，其语法格式为：

```
options {
    全局定义语句;
    …
    全局定义语句;
};
```

（2）区域配置部分：用于定义区域信息，包括 BIND 所负责解析的区域名称、区域类型以及区域数据文件名，其语法格式为：

```
zone  "域名"{
    区域信息定义语句;
    …
    区域信息定义语句;
};
```

在了解了 BIND 服务的主配置文件语法结构后，首先需要定义全局配置信息。

```
options {
        directory   "/var/named";
/*定义 BIND 服务的根目录,该目录将直接附加在配置文件中所使用的文件名前。*/
        dump-file  "data/cache_dump.db";
/*定义 BIND 服务器备份数据库文件的位置,该文件将附加在 directory 命令定义的根目录之
  后,完整路径为/var/named/data/cache_dump.db*/
        statistics-file   "data/named_stats.txt";
/*定义 BIND 服务器统计信息文件的位置,该文件将附加在 directory 命令定义的根目录之后,
  完整路径为/var/named/data/named_stats.txt*/
        memstatistics-file  "data/named_mem_stats.txt";
/*定义 BIND 服务器输出的内存使用统计信息文件的位置,该文件将附加在"directory"命令定
  义的根目录之后,完整路径为/var/named/data/named_mem_stats.txt*/
};
include  "/etc/rndc.key";        /*定义共享密码文件*/
```

以上是/etc/named.conf 文件的全局配置，在全局配置中要特别注意“directory "/var/named";”定义，今后在配置文件中所涉及的文件均位于该定义所声明的目录/var/named 之下。

完成全局配置之后，就需要声明 BIND 所负责解析的区域。在示例模型中，BIND 负责解析的区域为 xinya.com 和 shenghao.com。

在/etc/named.conf 文件中声明区域的方法为使用 zone 命令。

声明根区域，以便 BIND 可以向根域 DNS 进行迭代查询：

```
zone  "." {             /*定义 BIND 服务负责解析的区域名称*/
        type  hint;     /*type 命令用于定义该区域的类型,hint 表示的是根区域*/

        file  "named.ca";
                        /*file 命令用于定义根区域的区域数据文件,根区域的数据文件为
                        named.ca,该文件应附加在全局配置中 directory 命令定义的根目
                        录之后,完整路径为/var/named/data/named.ca*/
};
```

在根区域声明中，主要用到了两个命令。一个是 type 命令，用于定义区域类型；另一个是 file 命令，用于定义区域数据文件。注意，区域数据文件的存储位置是在全局配置中 directory 命令定义的根目录之后。

声明 BIND 负责解析的区域 xinya.com。注意，由于 dns.xinya.com 是 xinya.com 的主要 DNS 服务器，所以此处的区域数据类型应为 master，即主要区域。

```
zone "xinya.com" {        /*定义 BIND 服务负责解析的区域名称为 xinya.com*/
        type  master;     /*定义该区域的类型,即 BIND 服务是该区域的主要 DNS 服务器、
                          辅助 DNS 服务器还是惟高速缓存服务器,master 表示的是主要
                          区域,即主要 DNS 服务器*/
        file "xinya.com.zone";
                          /*定义 xinya.com 的区域数据文件。区域数据文件一般会以
                          .zone 作为文件扩展名,该文件也位于/var/named/目录下*/
};
```

在主要区域定义中，file 命令所定义的区域数据文件到目前为止是不存在的，因为并未建立过该文件，BIND 也不会自动建立该文件。区域数据文件是用户人工创建的，其中的内容也需要人工添加。在没有创建区域数据文件时是不能成功启动 named 服务的，因为 named 服务会发现缺少区域数据文件。

声明区域的方法很简单，对于 BIND 服务器所负责解析的另一区域 shenghao.com，读者可试着自行添加。下面给出/etc/named.conf 配置文件的最终结果。

```
options
{
        directory "/var/named"; // the default
        dump-file               "data/cache_dump.db";
        statistics-file         "data/named_stats.txt";
```

```
        memstatistics-file   "data/named_mem_stats.txt";
};

include "/etc/rndc.key";

zone "." {
    type hint;
    file "named.ca";
};
zone "xinya.com" {
    type master;
    file "xinya.com.zone";
};
zone "shenghao.com" {
    type master;
    file "shenghao.com.zone";
};
```

注意，此时还不能重新启动 BIND 服务使修改生效，因为刚刚声明的区域 xinya.com 与 shenghao.com 的区域数据文件还不存在。若重启 BIND 服务，则会收到“file not found”的错误提示。

```
[root@dns etc]# /etc/rc.d/init.d/named restart
停止 named:.                                                   [确定]
启动 named:
named 配置错误:
zone xinya.com/IN: loading master file xinya.com.zone: file not found
_default/shenghao.com/IN: file not found
zone shenghao.com/IN: loading master file shenghao.com.zone: file not
found
_default/shenghao.com/IN: file not found
                                                               [失败]
```

22.3.3 定义区域数据文件

区域数据文件中包含着本区域的相关数据信息，这些数据信息在 DNS 系统中称为记录。记录按其属性可分为 6 种类型。

（1）A 记录：主机记录，用于记录域名所对应的 IP 地址信息。

（2）CNAME 记录：别名记录，用于记录域名的别名，别名和域名将指向相同的 IP 地址。

（3）MX 记录：邮件交换器记录，用于定义该区域中的邮件服务器地址。

（4）PTR 记录：指针记录，是一种反向区域记录，用于记录 IP 地址所对应的域名。

（5）NS 记录：域名服务器记录，用于记录本区域的主要 DNS 服务器。

（6）SOA 记录：起始授权记录，用于定义辅助 DNS 服务器与主 DNS 服务器同步数

据时的操作。

这 6 种类型的记录不一定要全部存在于区域数据文件中，但是 SOA 记录与 NS 记录是必须要存在于区域数据文件中的。

创建区域数据文件，并添加 SOA 与 NS 记录。

区域数据文件存储于/var/named 目录下，文件名是在/etc/named.conf 文件中使用 file 命令定义的。此处首先创建 xinya.com 区域的数据文件/var/named/xinya.com.zone。

```
# vi  /var/named/xinya.com.zone
```

区域数据文件的结构从总体上可分为 4 个部分：

（1）文件头部定义：主要用于定义 TTL 值、管辖源等内容。

（2）SOA 定义：主要用于定义 SOA 记录。

（3）NS 定义：主要用于定义 NS 记录。

（4）其他记录定义：主要用于定义 A 记录、CNAME 记录、MX 记录和 PTR 记录等。

（1）文件头部定义：

```
$ttl  28800
```

DNS 客户端在收到 DNS 的查询结果相应信息后，可将该结果缓存至本机。但是应该缓存多长时间呢？这个时间值由 DNS 来定义，$ttl 就是用来定义客户端缓存的时间的。“$ttl 28800”表示客户端可缓存 28 800 s，在区域数据配置文件中，时间单位默认均为 s（秒）。也可以使用其他时间单位计量，如 8h（表示 8 小时）、2d（表示 2 天）和 1w（表示 1 星期），则此处可以表述为“$ttl 8h”。如果是内部使用的 DNS 服务器，且网络变化比较小，$ttl 可指定为 2d 等较长的时间。

（2）SOA 记录定义。

SOA 记录定义的语法结构为：

```
域名  IN  SOA  主要 DNS 服务器域名  DNS 服务器维护者邮件地址 (
               更新序列号
               更新间隔时间
               重试时间
               过期时间
               辅助服务器定义的缓存时间)
```

域名：是指本区域的域名，此处表明该 SOA 记录是属于哪个区域的。域名可以写域的全名（注意，要带有最后的根域标志“.”），也可以写为@。在 DNS 区域数据文件中，@表示本域。需要注意的一点是，当域名处保持为空时，则默认为本域。

IN：用于定义一条记录的类型，“IN SOA”表示该记录的记录类型为 SOA 记录。

主要 DNS 服务器域名：用于将主 DNS 服务器的完全合格域名通知辅助 DNS 服务器。注意，此处要写本域中主 DNS 服务器的完全合格域名。

DNS 服务器维护者邮件地址：当 DNS 服务器出现问题将通过邮件的形式通知服务器的维护者，而维护者的邮件地址就是在此处定义。标准的邮件地址是以@作为邮箱名

与邮件服务器域名的分隔符号的，但由于@符号在区域数据配置文件中有特殊的含义，因此此处的邮件地址使用“.”代替@。例如，原本的 root@xinya.com 地址会表述为 root.xinya.com。

（）括号内是辅助服务器与主服务器之间数据复制的选项，具体如下。

① 更新序列号：辅助服务器首次由主服务器复制数据时会将主服务器的更新序列号一并复制到本机，而后主服务器与辅助服务器分别维护该序列号。当主服务器内的数据记录发生变化时，该序列号会被更新。而辅助服务器向主服务器复制数据时会先比较该序列号，若序列号相同，则说明主服务器数据未发生变化，无须更新，否则说明主服务器发生了变化，需要辅助服务器更新数据。序列号的定义一般采用“年月日序号”这种编号形式，如 20110315001。

② 更新间隔时间：用于定义辅助服务器多长时间向主服务器更新一次数据，时间单位默认为 s，也可以使用 h、d、w 等时间单位来表示时间。

③ 重试时间：表示当辅助服务器向主服务器更新数据时，若主服务器未响应，则经过多长时间后重试。时间单位默认为 s，也可以使用 h、d、w 等时间单位来表示时间。

④ 过期时间：表示当辅助服务器向主服务器更新数据时，若主服务器未响应，则辅助服务器会定期重试。而经过多长时间后若重试还不成功，则辅助服务器会放弃重试。时间单位默认为 s，也可以使用 h、d、w 等时间单位来表示时间。注意，当辅助服务器放弃重试后，会将该辅助区域标示为无效，同时不会再向客户端计算机提供该区域的名称解析。

⑤ 辅助服务器定义的缓存时间：是指辅助服务器的客户端缓存查询结果的时间，若无$ttl 则以此时间为准。时间单位默认为 s，也可以使用 h、d、w 等时间单位来表示时间。

以下是 xingya.com 区域的 SOA 记录的示例。

```
xinya.com.  IN  SOA  dns.xinya.com  root.xinya.com. (
                2011070801
                10800
                3600
                604800
                38400)
```

（3）定义 NS 记录。

NS 记录定义的语法格式为：

```
域名  IN  NS  DNS 服务器的 FQDN
```

域名：是指本区域的域名，此处表明该 NS 记录是属于哪个区域的。域名可以写域的全名（注意，要带有最后的根域标志“.”），也可以写为@或保留为空。这一点与 SOA 记录相同。

IN NS：表示本条记录的记录类型为 NS 记录。

DNS 服务器的 FQDN：表示本区域的 DNS 服务器是哪台计算机，注意此处要使用 DNS 服务器的 FQDN（完全合格域名），即带有“.”根域标志的域名。

以下是 xingya.com 区域的 NS 记录的示例。

```
xinya.com.  IN  NS  dns.xinya.com.
```

在完成 SOA 记录与 NS 记录定义之后就可以进行其他记录的定义了。此时区域数据文件的内容为：

```
$ttl 28800
xinya.com.  IN  SOA  dns.xinya.com.  root.xinya.com. (
              2011070801
              10800
              3600
              604800
              38400)

xinya.com.  IN  NS  dns.xinya.com.
```

（4）定义 A 记录。

A 记录又称为主机记录，记录本区域（xinya.com）内都存在哪些主机，每台主机的 IP 地址是什么。对于 A 记录而言，首先要记录的就是 DNS 服务器这台主机 dns.xinya.com. 的 IP 地址，因为之前已经在 SOA 记录和 NS 记录中明确地声明了本域的主要 DNS 服务器是 dns.xinya.com.，而 dns.xinya.com.这台主机对应的 IP 地址是什么是要首先明确的。

A 记录定义的语法结构：

主机名或主机 FQDN IN A 主机 IP 地址

【示例】定义 dns.xinya.com.的 IP 地址为 192.168.1.105。

```
dns.xinya.com.  IN  A  192.168.1.105
```

也可以写成

```
dns  IN  A  192.168.1.105
```

【示例】定义 xinya.com 中的 www 主机，即 www.xinya.com.的 IP 地址为 203.6.5.8。

```
www  IN  A  203.6.5.8
```

也可以写成

```
www.xinya.com.  IN  A  203.6.5.8
```

事实上，DNS 服务的名称解析工作主要是依靠 A 记录的定义来完成的。当接收到域名解析请求时（如 www.xinya.com），DNS 服务程序会依据 xinya.com 区域数据文件中的 A 记录来查找到映射的 IP 地址，并将结果返回给用户的。

（5）定义 CNAME 记录。

CNAME 记录又称别名记录，该记录为 A 记录（主机记录）定义了别名，使得 DNS

客户端通过主机的别名也可以查询到主机的 IP 地址。例如，当为 www.xinya.com.主机设备别名为 ftp 后，则通过 ftp.xinya.com.也可以解析到 www.xinya.com.这台主机的 IP 地址。

CNAME 记录定义的语法结构如下：

```
主机别名  IN  CNAME  主机名或主机的 FQDN
```

【示例】定义 www.xinya.com.的别名为 ftp.xinya.com.。

```
ftp  IN  CNAME  www.xinya.com.
```

也可以写成：

```
ftp  IN  CNAME  www
```

（6）定义 MX 记录。

MX 记录又称为邮件交换器记录，该记录描述了本区域内的邮件服务器是哪台主机，以便其他邮件服务器能顺利定位本区域内的邮件服务器。对于存在有邮件服务器的域而言，必须在该域的 DNS 区域中定义邮件服务器的主机名。

MX 记录定义的语法结构如下：

```
域名  IN  MX  服务器优先级  邮件服务器主机的 FQDN
```

在 MX 记录的定义中，“域名”可以使用本域的实际域名，也可以使用@来表示。服务器优先级是指当本域中存在多台邮件服务器时，该服务器的优先接收邮件的顺序。优先级用数字来表示，数字越小，则优先级越高。在定义 MX 记录时还需注意，应使用邮件服务器的 FQDN（完全合格域名）来表示邮件服务器，而邮件服务器的 FQDN 所对应的 IP 地址是什么则需要使用 A 记录进行解析。

【示例】添加 xinya.com.的邮件服务器为 mail.xinya.com.，mail.xinya.com.主机的 IP 地址为 202.56.3.9。

```
//首先为 mail.xinya.com.添加 A 记录
mail.xinya.com.  IN  A  202.56.3.9
//添加 xinya.com.域的 MX 记录，优先级为 10
xinya.com  IN  MX  10  mail.xinya.com.
```

也可以写成

```
@  IN  MX  10  mail.xinya.com.
```

或

```
IN  MX  10  mail.xinya.com.
```

区域数据文件中应尽量包含网络中的所有主机，以便 DNS 能够提供对网络中的主机名的解析。以下是按前面的介绍所配置的 xinya.com.这个域的区域数据文件。

```
[root@dns named]# vi xinya.com.zone

$ttl 38400
xinya.com.  IN  SOA  dns.xinya.com.  root.xinya.com. (
            2011070801
            10800
            3600
            604800
            38400 )

xinya.com.  IN   NS      dns.xinya.com.

dns         IN   A       192.168.1.105
www         IN   A       192.168.1.105
ftp         IN   CNAME   www.xinya.com.
news        IN   CNAME   www
mail        IN   A       202.56.3.9
xinya.com   IN   MX  10  mail.xinya.com.
```

在本例中，除 xinya.com.域的区域数据文件外，shenghao.com.域的区域数据文件也需要人工添加，添加方法与 xinya.com.zone 相同。下面给出 shenghao.com 区域数据文件 shenghao.com.zone 的文件内容。

```
$ttl 38400
@          IN    SOA    dns.shenghao.com.    root.shenghao.com. (
             2011070801
             10800
             3600
             604800
             38400 )

@       IN      NS          dns.xinya.com.

dns     IN      A           192.168.2.1
www     IN      A           192.168.2.100
ftp     IN      CNAME       www.shenghao.com.
news    IN      CNAME       www
mail    IN      A           202.56.45.82
        IN      MX    10    mail.shenghao.com.
```

在完成数据文件的定义后，DNS 的解析信息就充足了。此时，可以重新启动 named 服务，以便服务可以重新加载配置文件和相关区域数据文件，使对服务所进行的修改生效。

在重新启动 named 服务时，若配置文件或数据文件存在语法错误，则 named 服务将不能顺利启动。需要修订相应的语法错误才能顺利启动 named 服务。

22.4　DNS 客户端的配置

在完成 DNS 服务器的配置并启动后，需要配置 DNS 的客户端。在以 Linux 操作系统作为 DNS 客户端的情况下，需要指定 DNS 服务器的 IP 地址信息，以便客户端主机可以定位 DNS 服务器的位置并提交查询。在 Linux 操作系统中，DNS 服务器 IP 地址信息的定义文件为/etc/resolv.conf，在配置文件中使用 nameserver 指令来指定 DNS 服务器的 IP 地址，最多可以指定 3 个 DNS 服务器的 IP 地址。

```
[root@dns named]# cat /etc/resolv.conf
; generated by /sbin/dhclient-script
nameserver 192.168.1.105
nameserver 219.141.140.10
```

DNS 客户端在获取 DNS 服务器的 IP 地址信息后，为了确定当前的 DNS 服务器能够正常提供 DNS 服务器，就需要使用 DNS 的测试工具了。

当安装了 bind-utils 工具包后，就可以使用其提供的 3 个 DNS 的测试工具来连接 DNS 服务器并进行测试。这 3 个 DNS 测试工具为 host、dig 和 nslookup。

22.4.1　host 命令

host 命令是一个简单的 DNS 服务测试工具，支持向 DNS 服务器进行正向名称解析、反向名称解析以及查询特定类型记录等功能。

1. 利用 host 命令进行正向名称解析测试

语法格式为：

host　完全合格域名

例如，测试 DNS 对 www.xinya.com 域名的解析：

```
[root@dns named]# host www.xinya.com
www.xinya.com has address 192.168.1.105
```

2. 利用 host 命令查询特定类型的记录

语法格式为：

host　-t　记录类型　域名

例如，查看 xinya.com 区域内的 SOA 记录内容与 NS 记录内容：

```
[root@dns ~]# host -t SOA xinya.com
xinya.com has SOA record dns.xinya.com. root.xinya.com. 2011070801 10800
3600 604800 38400
```

```
[root@dns ~]# host -t NS xinya.com
xinya.com name server dns.xinya.com.
```

3. 利用 host 命令查询特定主机的资源记录

语法格式为：

host -a 主机的完全合格域名

例如，查看 www.xinya.com 这套主机记录的详细信息：

```
[root@dns ~]# host -a www.xinya.com
Trying "www.xinya.com"
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 17956
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 1

;; QUESTION SECTION:
;www.xinya.com.                IN      ANY

;; ANSWER SECTION:
www.xinya.com.         38400   IN      A       192.168.1.105

;; AUTHORITY SECTION:
xinya.com.             38400   IN      NS      dns.xinya.com.

;; ADDITIONAL SECTION:
dns.xinya.com.         38400   IN      A       192.168.1.105

Received 81 bytes from 192.168.1.105#53 in 26 ms
```

4. 利用 host 命令查询整个区域的信息

语法格式为：

host -l 域名

例如，查询 xinya.com 域的全部信息：

```
[root@dns ~]# host -l xinya.com
xinya.com name server dns.xinya.com.
dns.xinya.com has address 192.168.1.105
mail.xinya.com has address 202.56.3.9
www.xinya.com has address 192.168.1.105
```

22.4.2 nslookup 命令

nslookup 命令是一个通用的 DNS 测试命令，该命令与 host 命令类似，都是用于测

试 DNS 服务器的记录信息的。

nslookup 命令是一个交互式命令，它提供一个命令行环境，在该命令行环境中可以对 DNS 服务器进行正向域名解析和反向域名解析等测试。

输入 nslookup 命令，将会得到一个 nslookup 命令行。

```
[root@dns ~]# nslookup
> www.xinya.com     ←输入主机名的 FQDN 进行正向域名解析测试
Server:        192.168.1.105
Address:       192.168.1.105#53

Name:   www.xinya.com
Address: 192.168.1.105
> set all     ←使用 set all 命令查看 DNS 服务器的设置
Default server: 192.168.1.105
Address: 192.168.1.105#53
Default server: 219.141.140.10
Address: 219.141.140.10#53

Set options:
  novc                  nodebug       nod2
  search                recurse
  timeout = 0           retry = 3     port = 53
  querytype = A         class = IN
  srchlist =
> set type=NS       ←定义 nklookup 待查看的记录类型为 NS 记录
> xinya.com         ←查看 xinya.com 区域中的 NS 记录信息
Server:             192.168.1.105
Address:            192.168.1.105#53
xinya.com           nameserver = dns.xinya.com.
> set type=SOA      ←定义 nklookup 待查看的记录类型为 SOA 记录
> xinya.com         ←查看 xinya.com 区域中的 SOA 记录信息
Server:             192.168.1.105
Address:            192.168.1.105#53
xinya.com
       origin = dns.xinya.com
       mail addr = root.xinya.com
       serial = 2011070801
       refresh = 10800
       retry = 3600
       expire = 604800
       minimum = 38400
> exit     ←退出 nslookup 命令环境
```

22.5　DNS 反向解析区域的配置

在 DNS 的区域结构中，除正向解析区域（将域名解析为 IP 地址）外，还存在反向解析区域。所谓反向解析区域是指将 IP 地址解析为域名，这种解析操作是为适应网络中特殊应用程序的需要。下面介绍如何配置反向解析。

同正向解析区域一样，也要在 DNS 主配置文件中加入要管理的反向区域。反向区域是以倒序的 IP 网段加.in-addr.apra 的形式来进行命名的，如 1.168.192.in-addr.arpa。

```
[root@dns ~]# vi /etc/named.conf
zone "." {

   type hint;
   file "named.ca";
};

zone "xinya.com" {

   type master;
   file "xinya.com.zone";
};

zone "shenghao.com" {


   type master;
   file "shenghao.com.zone";
};
zone "1.168.192.in-addr.arpa" IN {  ←定义反向解析区域
   type master;
   file "192.168.1.zone";
};
```

在主配置文件中，区域的类型依然是 master（主 DNS 服务器）类型，而区域文件的文件名在这里依然可以随意命名，但是为了方便以后的管理，建议使用网段加.zone 的方式进行命名。如上例中的 192.168.1.zone。

当主配置文件配置完成后，下面就要编写反向区域文件。同正向区域文件一样，反向区域文件也放置在/var/named/目录下。区域文件名要与主配置文件中指定的文件名相同。

反向区域文件的书写格式与正向区域文件类似，区别在于正向区域文件是将域名映射到 IP 上，而反向区域文件则是将 IP 匹配到正向区域文件中所映射的域名上，其记录类型为 PTR 指针记录。

```
$ttl 38400
@           IN    SOA    dns.xinya.com.    root.xinya.com. (
            2011070801
            10800
            3600
            604800
            38400 )

            IN     NS     dns.xinya.com.

110     IN    PTR    dns.xinya.com.
111     IN    PTR    bbs.xinya.com.
112.1.168.192.in-addr.arpa    IN    PTR   mail.xinya.com.
```

PTR 记录称为反向资源指针记录。PTR 记录的作用就是将 IP 地址匹配到正向区域文件中所映射的域名上。

需要注意的是，在反向区域文件中不可以出现 A 记录；反之，在正向区域文件当中也不可以出现 PTR 记录。

需要说明的是，在反向区域文件中@的意义发生了变化，在上例中不再代表正向区域文件中的 xinya.com.，是代表反向区域的 1.168.192.in-addr.arpa。

反向区域文件配置完成后，解析结果如下：

```
[root@dns ~]# nslookup
> 192.168.1.110
Server:         192.168.1.110
Address:        192.168.1.110#53

112.1.168.192.in-addr.arpa      name = www.xinya.com.
112.1.168.192.in-addr.arpa      name = dns.xinya.com.
```

第23章　Web 服务的配置与应用

随着 Internet 上 Web 服务的发展，配置和管理 Web 服务器成为 Internet 必不可少的工作。本章主要介绍如何利用 Apache 服务器软件来搭建作为 WWW 应用基础架构的 Web 服务器。

23.1　Web 服务简介

Web 服务是 Internet 中最为重要的应用，是 WWW 网络中诸多应用的基本平台，并通过超级链接（Hypertext）的方式将信息通过 Internet 传递到世界各处。

23.1.1　HTTP 协议

WWW 的目的就是将地理位置不同的计算机连接在一起，以便能方便地获取信息、实现资源共享。而信息的共享方式是利用 HHTP 协议，即超文本传输协议来实现的。

HTTP 是应用层协议，主要用于分布式、协作的信息系统。HTTP 协议是通用的、无状态的，其系统的建设和传输的数据无关。HTTP 也是面向对象的协议，可以用于各种任务，包括名字服务、分布式对象管理、请求方法的扩展和命令等。

在 Internet 上，HTTP 通信往往发生在 TCP/IP 连接上，其默认的端口为 80，也可以使用其他的端口。

23.1.2　Web 服务

Web 服务采用客户机/服务器架构模式。

（1）客户机运行 WWW 服务的客户程序——浏览器，以提供良好、统一的用户界面。浏览器的作用主要是解释和显示 Web 页面，响应用户的输入请求，并通过 HTTP 协议将用户请求传递给 Web 服务器。

（2）Web 服务器端运行服务器程序，它最基本的功能是侦听和响应客户端的 HTTP 请求，向客户发出请求结果信息。

23.1.3　Web 服务的工作原理

Web 服务使用 HTTP（超文本传输协议）进行数据通信，该协议是一个在 TCP/IP 协议基础上的应用程序级协议，它的具体通信过程如图 23.1 所示。

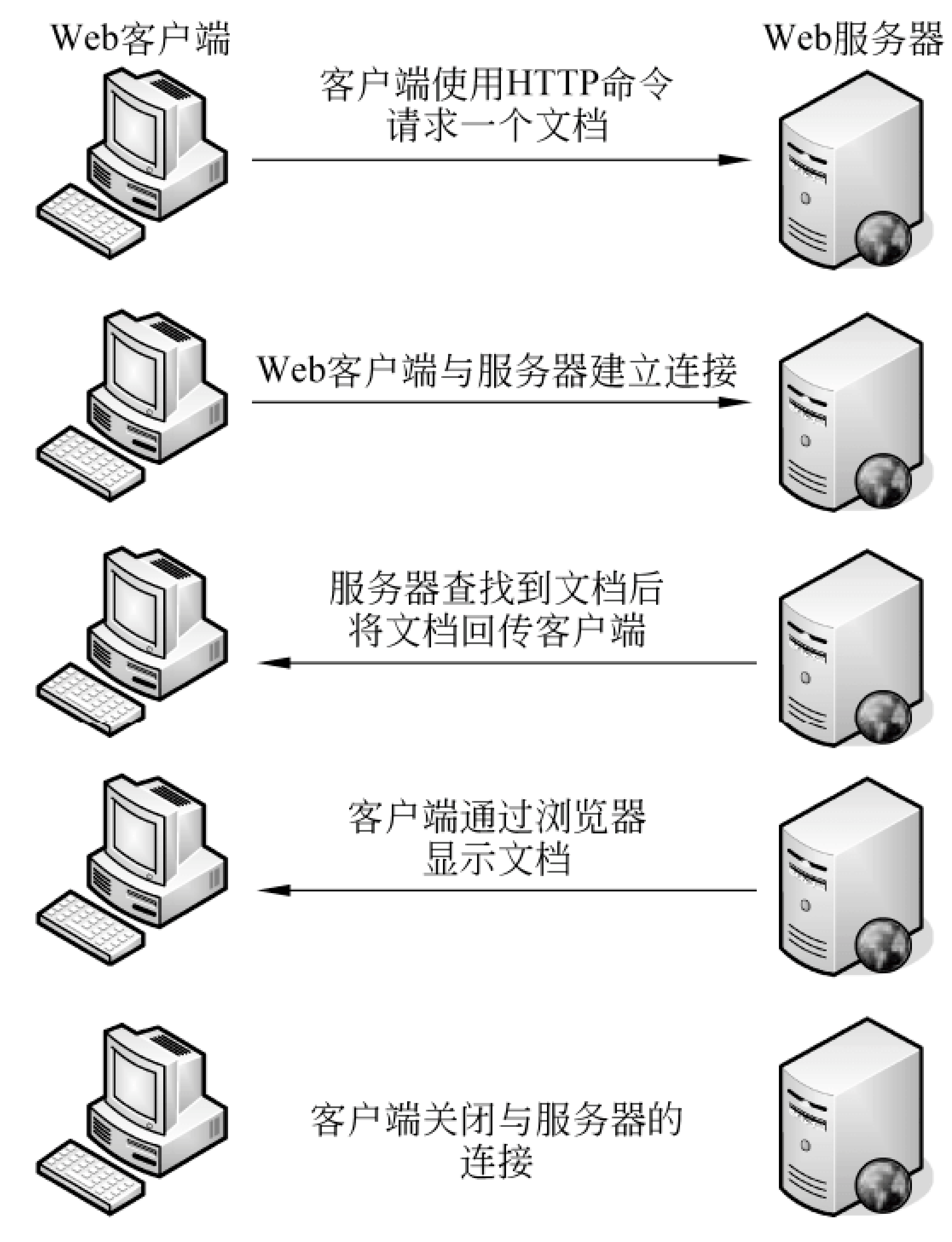


图 23.1　HTTP 协议的通信过程

（1）Web 客户端程序——浏览器使用 HTTP 命令向一个特定的服务器发出 Web 页面请求。

（2）若该项服务器在特定的端口（通常是 TCP 80 端口）处接收到 Web 页面请求，就发送一个应答，并在客户和服务器之间建立连接。

（3）Web 服务器查找客户端所需文档。若 Web 服务器查找到客户端所请求的文档，就会将文档传送给 Web 浏览器。若该文档不存在，则服务器会发送一个相应的错误提示文档给客户端。

（4）Web 浏览器接收到文档后，就将它显示出来。

（5）当客户端浏览完成后，断开与服务器的连接。

23.2　Apache 服务器

当前 Linux 操作系统多使用 Apache 服务器程序来实现 Web 服务，Apache 服务器程序是一种高效、稳定、易扩展的开源应用软件，是当前主流的 Web 服务器实现程序。

23.3　Apache 服务的安装

当前的 Linux 发行版大都默认安装了 Apache 服务器。由于目前 Apache 被重新命名为 httpd，因此可使用“rpm　-q　httpd”命令检查系统是否安装了 Apache。

```
[root@dns named]# rpm -q httpd
httpd-2.2.3-43.el5.centos
```

如果系统中没有安装 Apache，则需要手工安装该应用程序。在 CentOS 中，Apache 软件包的存储位置为安装光盘的 CentOS 目录下，软件包名为 httpd-2.2.3-43.el5.centos.i386.rpm，可以使用 rpm 命令进行安装。

```
[root@dns CentOS]# rpm -ivh httpd-2.2.3-43.el5.centos.i386.rpm
warning: httpd-2.2.3-43.el5.centos.i386.rpm: Header V3 DSA signature:
NOKEY, key ID e8562897
Preparing...          ########################################### [100%]
```

将 Apache 软件包安装好之后，得到一个名称为 httpd 的守护进程，该进程就是 Apache 服务的守护进程。与其他服务器进程一样，httpd 服务也是通过位于/etc/rc.d/init.d/目录下的 httpd 服务控制脚本来控制的。也可以使用 service 命令来直接启动、关闭和重新启动该服务。

```
service  httpd  start        ←启动 httpd 服务
service  httpd  stop         ←停止 httpd 服务
service  httpd  restart      ←重新启动 httpd 服务
```

启动 httpd 服务后，为测试 Apache 的服务状态，可在浏览器的地址栏中输入 Apache 服务器所在的 IP 地址进行访问。如果出现 Apache 的测试页面，则表示 Web 服务安装正确并运行正常，如图 23.2 所示。

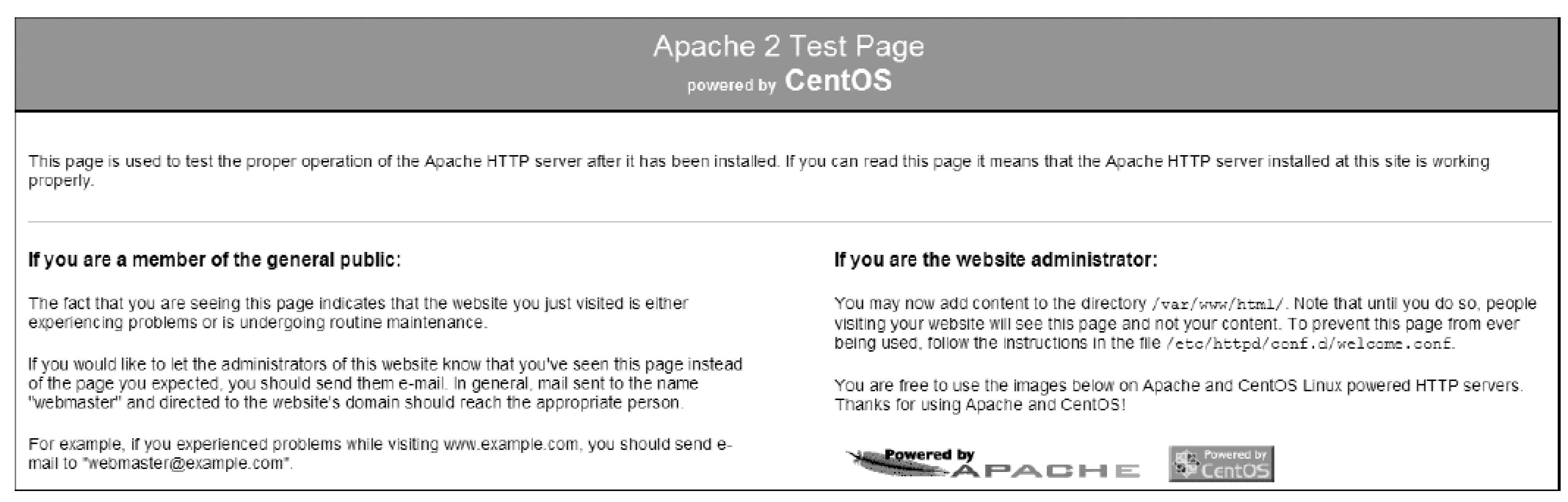


图 23.2　Apache 测试页面

23.4　Apache 服务器的配置

Apache 服务的配置是通过编辑 Apache 的主配置文件 httpd.conf 来实现的。该文件的位置随着安装方式的不同而不同，如果使用 RPM 软件包的方式安装，该文件通常存放在/etc/httpd/conf/目录下。如果使用编译源代码的方式安装，该文件通常存放在 Apache 安装目录的 conf 子目录下。由于是采用 RPM 包格式安装，所以配置文件应为/etc/httpd/conf/httpd.conf。

23.4.1　httpd.conf 文件的格式

/etc/httpd/conf/httpd.conf 配置文件主要 3 个部分组成。

（1）全局环境配置（Section 1：Global Environment）：用于定义整个 Apache 服务器运行的环境。

（2）主服务器配置（Section 2：Main Server configuration）：用于定义 Apache 主服务器特性。

（3）虚拟主机配置（Section 3：Virtual Hosts）：用于定义 Apache 虚拟主机。

配置文件的每个部分都有相应的配置语句，文件的语法结构为：

```
配置参数名称　参数值
```

配置语句可以放在文件中的任何地方，但为了增强文件的可读性，最好将配置语句放在相应的部分。

在 httpd.conf 中每行包含一条语句，行末使用反斜杠可以换行，但是反斜杠与下一行中间不能有任何其他字符，包括空白。httpd.conf 的配置语句除了选项的名称外，所有选项指令均不区分大小写，可以在每一行前用#号表示注释。

在默认的 httpd.conf 文件中，每个配置语句和参数都有详细的解释，建议初学者在还不熟悉配置方法时，先使用 Apache 默认的 httpd.conf 文件作为模板进行设置，而且在修改之前先做好备份，以便做了错误的修改后能够还原。

23.4.2　Web 服务的基本配置

1. 定义主目录的路径

Apache 服务器主目录的默认路径位于/var/www/html 目录，可以将需要发布的网页放在该目录下。也可以将主目录的路径修改为其他目录，以方便管理和使用。

在配置文件中定义主目录位置的语句为：

```
DocumentRoot "/var/www/html"     ←Apache 服务的主目录位置为/var/www/html
```

当需要将主目录改为/var/apache/目录时可直接修改 DocumentRoot 命令的参数值：

```
DocumentRoot "/var/apache"       ←定义 Apache 服务的主目录为/var/apache
```

2. 定义默认文档

默认文档是指在 Web 浏览器中输入 Web 站点的 IP 地址或域名即显示出来的 Web 页面（即在 URL 中没有指定要访问的页面），也就是通常所说的主页。Apache 的默认文档名为 index.html。默认文档名由 DirectoryIndex 语句进行定义，可以将 DirectoryIndex 语句中的默认文档名修改为其他文件名：

```
DirectoryIndex  index.html  index.html.var
```

如果有多个文件名，每个文件名之间须用空格分隔。Apache 会根据文件名的先后顺

序查找在“主目录”列表中指定的文件名，如能找到第一个则调用第一个，否则再查找并调用第二个，依此类推。如下所示：

```
DirectoryIndex  index.html  index.html.var  index.htm
```

3. 定义 Apache 服务所监听的 IP 地址和端口号

Apache 默认会根据 Listen 语句的定义监听本机所有可用 IP 地址上的 TCP 80 端口。可以使用多个 Listen 语句，以便在多个地址和端口上监听请求。

```
Listen 80                    ←定义 Apache 服务默认监听本机所有 IP 地址上的 80 端口
```

如果想要同时监听多个 IP 上的不同端口，操作如下：

```
Listen 192.168.1.110:80
Listen 192.168.1.210:8080
```

如果 Apache 监听的 TCP 端口号是 80 以外的其他端口，当用户在请求该服务器时，需要在访问时指定其服务器监听的端口才可以正常访问。例如，要访问 www.xinya.com，该站点的端口号为 8080，在访问时要在地址栏中输入 www.xinya.com:8080。

4. 定义相对根目录的路径

相对根目录通常是 Apache 存放配置文件和日志文件的地方。在默认的情况下，相对根目录是/etc/httpd，它一般包含 conf 和 logs 子目录。设置如下：

```
ServerRoot  "/etc/httpd"
```

5. 定义日志文件

日志文件可以说是网络管理员最好的帮手，分析日志文件是每个网络管理员必不可少的工作，通过日志文件可以了解 Apache 的运行情况、出错原因和安全等问题。

1）错误日志

错误日志记录了 Apache 在启动或运行时发生的错误，所以当 Apache 出错时，应该先检查这个日志。通常错误日志的文件名为 error_log，错误日志存放的位置和文件名可以通过 ErrorLog 参数设置，如下所示：

```
ErrorLog  logs/error_log
```

这里要提醒的是，如果日志文件存放的路径不是以/开头，则意味着该路径是相对于 ServerRoot 目录的相对路径。

2）访问日志

访问日志记录了客户端所有的访问信息，通过分析访问日志可以知道客户机什么时间访问了网站的什么文件等信息。通常访问日志的文件名为 access_log。访问日志存放的位置和文件名可以通过 CustomLog 参数设置。如下所示：

```
CustomLog  logs/access_log combined
```

上面的语句最后的combined指明日志使用的格式，在这个位置可以使用common或combined（使用自定义的名称也可以）。common是指使用Web服务器普遍采用的“普通标准”格式（Common Log Format），这种格式可以被许多日志分析程序所识别。combined是指使用“组合记录”格式（Combined Log Format），其实combined和common格式基本相同，只是多了“引用页”和“浏览器识别”信息而已。common和combined格式由LogFormat语句进行定义，如下所示：

```
LogFormat  "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\""
combined
LogFormat  "%h  %l  %u  %t  \"%r\"  %>s  %b  common
```

6. 定义网络管理员的E-mail地址

当客户端访问服务器发生错误时，服务器通常会向客户端返回错误提示网页，为了方便解决错误，在这个网页中通常包含有管理员的E-mail地址。可以使用ServerAdmin语句来设置管理员的E-mail地址，如下所示：

```
ServerAdmin  xinyaadmin@xinya.com
```

7. 定义服务器主机名称

为了方便Apache识别服务器自身的信息，可以使用ServerName语句来设置服务器的主机名称。在ServerName语句中，如果服务器有域名，则填入服务器的域名；如果没有域名，则填入服务器的IP地址。如下所示：

```
ServerName  192.168.1.110
```

8. 定义默认字符集

AddDefaultCharset 选项定义了服务器返回给客户机的默认字符集。由于 UTF-8 是Apache的默认字符集，因此当客户访问服务器的中文网页时会出现乱码，解决的办法是将语句“AddDefaultCharset UTF-8”改为“AddDefaultCharset GB2312”或“#AddDefaultCharset UTF-8”，然后重新启动Apache服务，中文网页就能正常显示了。选项格式如下：

```
AddDefaultCharset  UTF-8
```

在修改完默认字符集后，如果以前访问过中文网页，应清空Web浏览器的缓存后再测试，否则会由于缓存的原因造成虽然修改了默认字符集，但Web浏览器还是显示乱码。

23.5 Web 服务的启动和停止

23.5.1 启动 Web 服务

Apache 服务配置完成后，启动测试服务配置是否成功，具体命令为：

```
[root@dns conf]# /etc/rc.d/init.d/httpd start
启动 httpd:                                                    [确定]
```

23.5.2 停止 Web 服务

停止 Apache 服务，具体命令为：

```
[root@dns conf]# /etc/rc.d/init.d/httpd stop
停止 httpd:                                                    [确定]
```

23.5.3 重新启动 Web 服务

重新启动 Apache 服务，具体命令为：

```
[root@dns conf]# /etc/rc.d/init.d/httpd restart
停止 httpd:                                                    [确定]
启动 httpd:                                                    [确定]
```

第24章　远程管理工具的管理与使用

远程管理也可以称为远程控制，是指由一台计算机（客户机）通过网络连接到另一台远程计算机（服务器），进而可以使用远程服务器上的软、硬件资源，达到控制远程服务器的目的。远程管理作为系统管理中的一种重要手段在人机分离的环境中是经常被使用的。所谓“人机分离”是指网络中的服务器计算机通常位于专门的机房或 ISP 托管机房，管理者与服务器在地理位置上是分离的，因此要借助远程管理工具对服务器进行管理。

在 Windows 平台上，可以通过终端服务或第三方软件（如 PcAnywhere）实现对服务器的远程管理。而在 Linux 平台下，通常采用 Telnet、SSH 和 VNC 等工具实现远程管理。其中 Telnet 由于在网络中使用明文数据传输，导致数据安全性下降，因而已经很少被使用了。而 VNC 尽管支持 GUI 界面，但是由于网络开销比较大，所以也不常用。本章主要介绍 SSH 远程管理工具，SSH 具有数据加密传输、网络开销小以及应用平台广泛等特点，是远程管理中最常见的控制工具。

24.1　SSH 服务概述

当前远程管理环境中最常使用的是 SSH（Secure Shell）。SSH 是一个可在应用程序中提供安全通信的协议，通过 SSH 可以安全地进行网络数据传输，这得益于 SSH 采用的非对称加密体系，即对所有待传输的数据进行加密，保证数据在传输时不被恶意破坏、泄露和篡改。SSH 支持多种加密和认证方式，解决了传输中数据加密和身份认证的问题，能有效防止网络嗅探和 IP 欺骗等攻击。

目前 SSH 协议已经经历了 SSH1 和 SSH2 两个版本，这两个版本采用不同的实现方式与协议，因此二者互不兼容。SSH2 在安全性、效率性与应用功能上都比 SSH1 更有优势，所以目前得到了更广泛的使用。

24.2　SSH 服务的安装

在 Linux 操作系统中，通常使用 OpenSSH 程序来实现 SSH 机制，OpenSSH 是一个 GPL 协议软件，它同时支持 SSH1 和 SSH2 这两种协议。

OpenSSH 是由 OpenBSD project 开发与维护的，其官方网站是 http://www.openssh.

org/，该站点包含了有关 OpenSSH 最新的错误修复和更新。目前大多数 Linux 发行版都默认提供了 OpenSSH 应用程序，CentOS 5 对 OpenSSH 采用的是默认安装的策略。可使用“rpm -q”命令来查看系统默认安装的 OpenSSH 服务的版本。

```
[root@dns CentOS]# rpm -qa | grep openssh
openssh-askpass-4.3p2-41.el5        ←OpenSSH 密码文件
openssh-clients-4.3p2-41.el5        ←OpenSSH 客户端程序
openssh-server-4.3p2-41.el5         ←OpenSSH 服务器程序
```

如果系统当中并未安装 OpenSSH，则可通过安装光盘获得 OpenSSH 安装包并安装。

```
[root@dns CentOS]# rpm -ivh openssh-server-4.3p2-41.el5.i386.rpm
warning: openssh-server-4.3p2-41.el5.i386.rpm: Header V3 DSA signature:
NOKEY, key ID e8562897
Preparing...           ########################################### [100%]
   1.openssh-server    ########################################### [100%]
```

OpenSSH 软件包由两部分组成，一部分是服务器软件包 openssh-server，另一部分是客户端软件包 openssh-clients，它们分别打包在两个不同的 RPM 软件包中，此处可以先安装 openssh-server，待用到 OpenSSH 的客户端程序时再安装 openssh-clients。

24.3 SSH 服务的配置

SSH 服务是通过配置文件/etc/ssh/sshd_config 进行管理和配置的，/etc/ssh/sshd_config 文件默认为 OpenSSH 提供了一套可以正常运行的配置，同时，该配置文件还提供了大量的配置选项供用户按需配置 OpenSSH，这些额外的选项一般都使用#符号注释掉了，用户可以按需开启这些选项。此处介绍 OpenSSH 的一些常用选项。

1. 定义 SSH 服务监听的端口号

SSH 服务所监听的端口号使用 Port 选项进行定义，SSH 服务默认使用的端口号是 22。

```
…
#   IdentityFile ~/.ssh/id_rsa
#   IdentityFile ~/.ssh/id_dsa
    Port 22
#   Protocol 2,1
#   Cipher 3des
…
```

2. 定义使用 SSH 协议的顺序

Protocol 选项定义了 SSH 服务器使用 SSH 协议的顺序，默认先使用 SSH2（用 2 表

示）协议。如果不成功则使用 SSH1（用 1 表示）协议。为了安全起见，可以设置只使用 SSH2 协议。

```
…
#   IdentityFile ~/.ssh/id_dsa
    Port 22
    Protocol 2,1
#   Cipher 3des
#   Ciphers   aes128-cbc,3des-cbc,blowfish-cbc,cast128-cbc,arcfour,
aes192-cbc,aes256-cbc
…
```

3. 定义 SSH 服务器绑定的 IP 地址

ListenAddress 选项定义了 SSH 服务器绑定的 IP 地址，默认绑定服务器所有可用的 IP 地址“ListenAddress 0.0.0.0”。若需要指定 SSH 服务所绑定的 IP 地址，可修改 ListenAddress 的定义。例如，要求 SSH 服务绑定 192.168.1.105 这个 IP 地址，即只在该 IP 地址上提供 OpenSSH 服务，可以使用“ListenAddress 192.168.1.105”，如下所示：

```
#   TunnelDevice any:any
#   PermitLocalCommand no
    ListenAddress 192.168.1.105
    Host *
    GSSAPIAuthentication yes
```

4. 定义是否允许 root 管理员登录

PermitRootLogin 选项定义了是否允许 root 管理员通过 SSH 登录，默认允许管理登录“PermitRootLogin yes”，也可以在“PermitRootLogin”选项中使用 no 表示不允许管理员利用 SSH 登录。

```
#   TunnelDevice any:any
#   PermitLocalCommand no
    PermitRootLogin  yes | no
    ListenAddress 192.168.1.105
    Host *
```

5. 定义是否允许空密码用户登录

PermitEmptyPasswords 选项定义了是否允许空密码的用户登录。为了保证服务器的安全，应该禁止这些用户登录，默认是禁止空密码用户登录。

```
PermitEmptyPasswords  no
```

当使用“PermitEmptyPasswords yes”时表示允许用户使用空密码登录。

6. 定义是否使用口令认证方式

PasswordAuthentication 选项定义了是否使用口令认证方式。如果准备使用公钥认证方式，可以将其设置为 no。

```
PasswordAuthentication  yes
```

24.4 SSH 服务的启动和停止

SSH 服务的守护进程名称为 sshd，可以通过该服务器的启动控制脚本/etc/rc.d/inid.d/sshd 来控制 SSH 服务，也可以通过 CentOS 中的 service 命令来控制 SSH 服务。

1. 启动 SSH 服务

启动 SSH 服务的命令如下：

```
[root@dns CentOS]# /etc/rc.d/init.d/sshd start
启动 sshd:                                            [确定]
```

2. 停止 SSH 服务

停止 SSH 服务的命令如下：

```
[root@dns CentOS]# /etc/rc.d/init.d/sshd stop
停止 sshd:                                            [确定]
```

3. 重新启动 SSH 服务

重新启动 SSH 服务的命令如下：

```
[root@dns CentOS]# /etc/rc.d/init.d/sshd restart
停止 sshd:                                            [确定]
启动 sshd:                                            [确定]
```

24.5 SSH 客户端的使用

24.5.1 Windows 平台

1. 获取 PuTTY 程序

Windows 下有许多 SSH 的客户端程序，常用的是一种名为 PuTTY 的免费 SSH 客户端程序，它不但小巧好用，而且是一款无须安装的绿色软件。当前 PuTTY 的最新版本是 0.60。其官方维护站点为：

http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html

2. 连接 SSH 服务器

运行 putty.exe 文件，在 PuTTY 程序主界面中的 Host Name 文本框中输入服务器的 IP 地址或域名，在 Connection type 下选择 SSH 单选按钮，然后单击 Open 按钮连接即可，如图 24.1 所示。

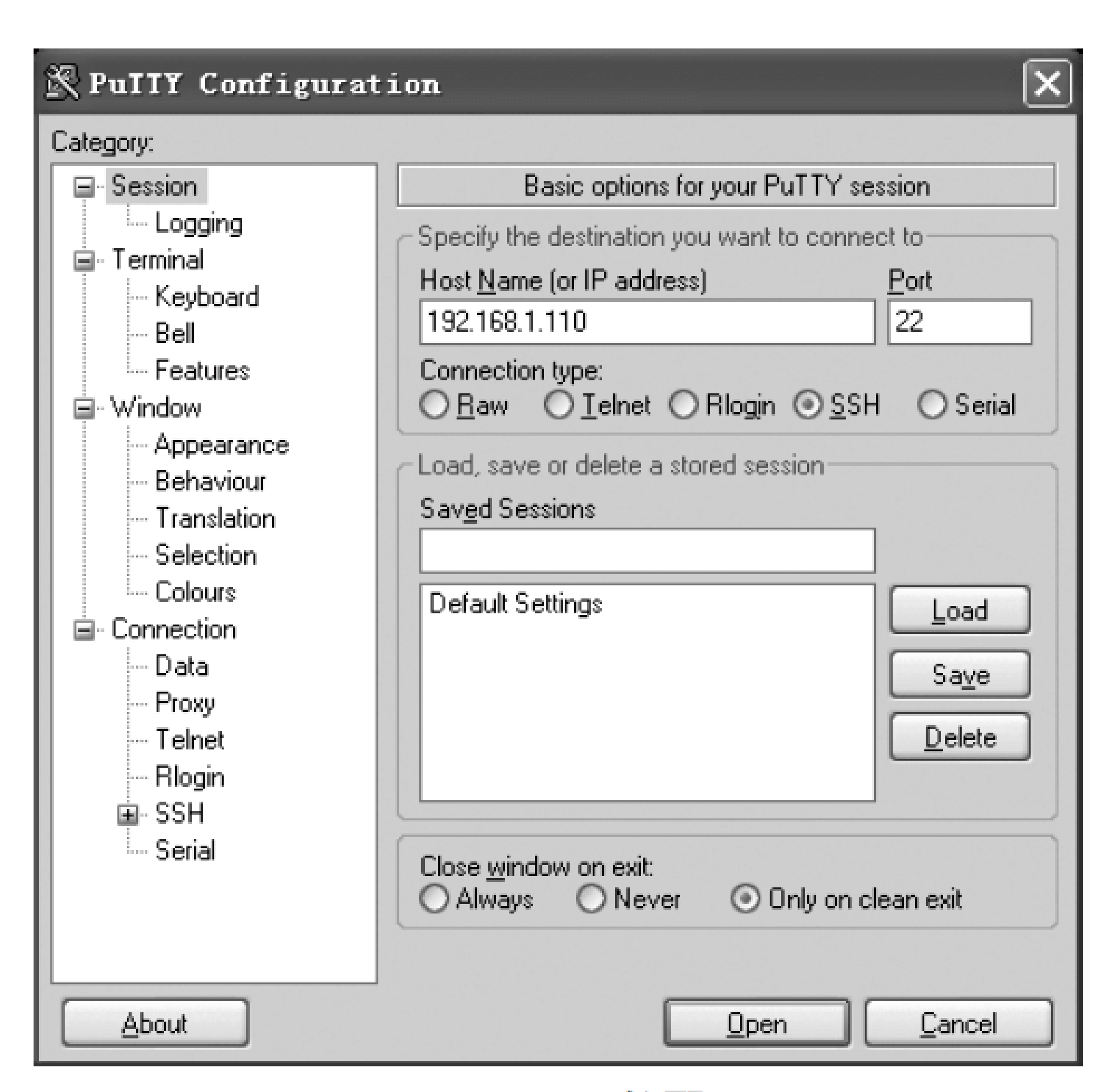


图 24.1 PuTTY 的配置界面

如果是第一次连接到某台服务器，由于服务器公钥还没有在注册表中缓存，PuTTY 程序会出现警告信息并显示服务器的指纹信息，如图 24.2 所示。



图 24.2 PuTTY 公钥安全提示

该警告信息主要是要求用户核对公钥是否与需要连接的服务器一致，以防止欺骗技术。如果公钥得到用户认可，可以单击“是（Y）”按钮来确认，PuTTY 程序同时会将服务器公钥缓存在注册表中，下次连接时就不会再出现提示了。

3. 在行程系统上工作

如果 PuTTY 成功地连接到 SSH 服务器，就会显示登录信息并提示用户输入用户名

和密码，如果用户名和密码输入正确，就能成功登录并在远程系统上工作了，如图 24.3 所示。

```
root@xinya:~
login as: root
root@192.168.1.110's password:
Last login: Mon Aug 23 19:00:59 2010
[root@xinya ~]# ls
anaconda-ks.cfg  Desktop  install.log  install.log.syslog
```

图 24.3 PuTTY 应用界面

24.5.2 Linux 平台

在 Linux 平台下可以使用 OpenSSH 客户端程序 openssh-clients 来连接 SSH 服务器。CentOS 5 默认已将 SSH 客户程序安装在系统上。可使用下面的命令检查系统是否已经安装了 OpenSSH 的客户端程序。

```
[root@dns CentOS]# rpm -q openssh-clients
openssh-clients-4.3p2-41.el5
```

如果系统中没有安装该软件包，则可将光盘放入光驱中，将光驱挂载到系统中，到挂载目录下的 Server 目录中去查找并安装这个软件包。

安装好 openssh-clients 程序后，可以直接使用 ssh 命令登录到 SSH 服务器，格式如下：

```
ssh  服务器的 IP 地址或域名
```

与 PuTTY 程序类似，openssh-clients 程序在第一次连接到某台服务器时也会出现警告信息，可以输入“yes”确认，程序会将服务器公钥缓存在当前用户主目录下.ssh 子目录中的 known_hosts 文件里（如/root/.ssh/known_hosts），下次连接时就不会出现提示了。如果成功地连接到 SSH 服务器，就会显示登录信息并提示用户输入用户名和密码。如果用户名和密码输入正确，就能成功登录并在远程系统上工作了。

```
[root@dns CentOS]# ssh 192.168.1.105
The authenticity of host '192.168.1.105 (192.168.1.105)' can't be established.
RSA key fingerprint is 65:bc:b7:39:d1:32:69:7d:14:42:1e:8d:69:15:56:e4.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.1.105' (RSA) to the list of known hosts.
root@192.168.1.105's password:
Last login: Fri Jul 29 14:18:28 2011 from www.xinya.com
[root@dns ~]#
```

OpenSSH 服务提供了 sftp 功能，使用它能安全、方便地传输机密文件。OpenSSH 服务启动后就能提供 sftp 功能，而且 sftp 的命令格式与 FTP 相同，操作起来非常方便。在 Windows 下可以使用 PuTTY 提供的 psftp 作为客户程序，在 Linux 下可以使用 openssh-clients 提供的 sftp 作为客户程序。

24.6 使用非对称加密认证

24.6.1 非对称加密体系结构

非对称加密体系结构理论基础是非对称性算法，非对称性算法中存在两种加密的密钥，一种称为公钥（Public Key），另一种称为私钥（Private Key），在非对称加密算法中，使用公钥加密的数据必须利用私钥解密；同理，使用私钥加密的数据必须利用公钥解密，如图 24.4 所示。在非对称加密体系结构中，对象同时具有公钥与私钥，且公钥是对外开放的，任何用户与应用程序均可向对象申请公钥，而私钥却是仅保留给对象自行使用的。其基本含义是：任何用户均可以利用对象的公钥加密数据，但只有对象本人可以利用其私钥解密该数据。

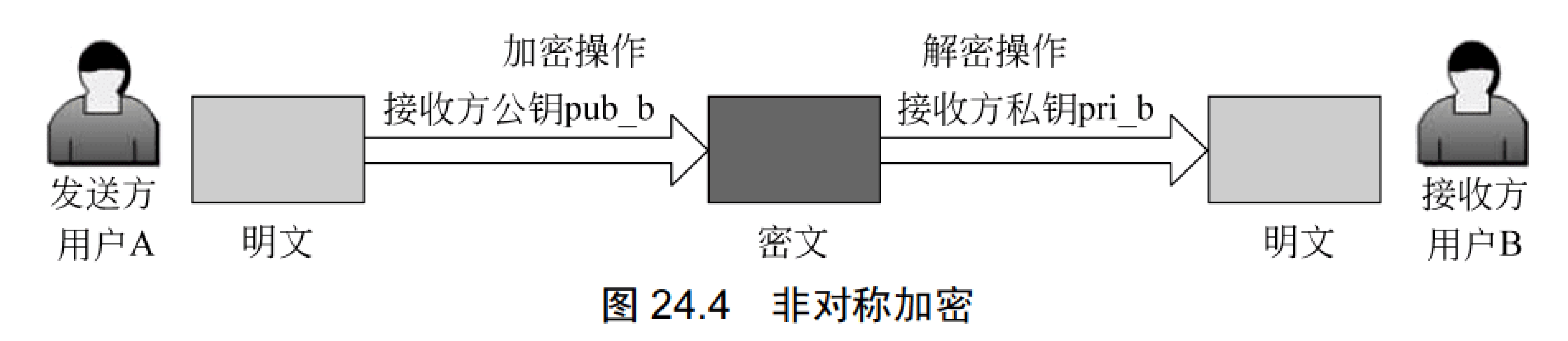


图 24.4 非对称加密

以图 24.4 为例，用户 A 要发送数据给用户 B，主要经过以下步骤进行加密：

（1）用户 B 通过各种方式（如网络）向外公开他的公钥 pub_b。

（2）用户 A 得到用户 B 的公钥 pub_b，并使用 pub_b 对信息加密发送给用户 B。

（3）用户 B 在收到密文后，使用其私钥 pri_b 就能完成解密。

从上述步骤可以看出，如果密文在传输的过程中被黑客截取，由于黑客没有用户 B 的私钥 pri_b，因此不能对其解密。

与传统的对称型加密方法相比，公钥加密体系具有密钥分配与管理简单、可实现数字签名和身份验证的优点。目前，常用的非对称算法有 RSA 和 DSA。

24.6.2 非对称加密认证的原理

首先由用户生成一对密钥（包含公钥与私钥），然后将公钥保存在 SSH 服务器用户主目录下.ssh 子目录中的 authorized_keys 文件里（如/root/.ssh/authorized_keys），私钥保存在本地计算机中。当用户登录时，服务器检查 authorized_keys 文件的公钥是否与用户的私钥对应，如果相符则允许用户登录，否则拒绝用户的登录请求，由于私钥只保存在用户的本地计算机中，因此入侵者就算得到用户的口令，也不能登录到服务器。

24.6.3 在服务器启用公钥认证

编辑文件/etc/ssh/sshd_config，找到语句“PasswordAuthentication yes”，并将语句改为“PasswordAuthentication no”。

```
23 #   RhostsRSAAuthentication no
24 #   RSAAuthentication yes
```

```
    PasswordAuthentication no
#   HostbasedAuthentication no
#   BatchMode no
```

24.6.4 在 PuTTY 程序使用公钥认证

1. 获取 PuTTYgen 程序

为了生成密钥，还要到以下网址下载产生密钥的程序 puttygen.exe：

```
http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html
```

PuTTYgen 是一套可以产生密钥的工具，它可生成 RSA 和 DSA 的密钥，产生的公钥和私钥可以用于 PuTTY、PSCP、Plink 和 Pageant。

2. 产生密钥

运行下载的 puttygen.exe 文件，在 PuTTYgen 程序主界面中的“Type of key to generate:”下选择加密的算法，在“Number of bits in a generated key:”后的文本框中输入加密的位数，推荐使用默认的 1024 位 SSH2 RSA 加密，然后单击 Generate 按钮开始生成密钥，如图 24.5 所示。

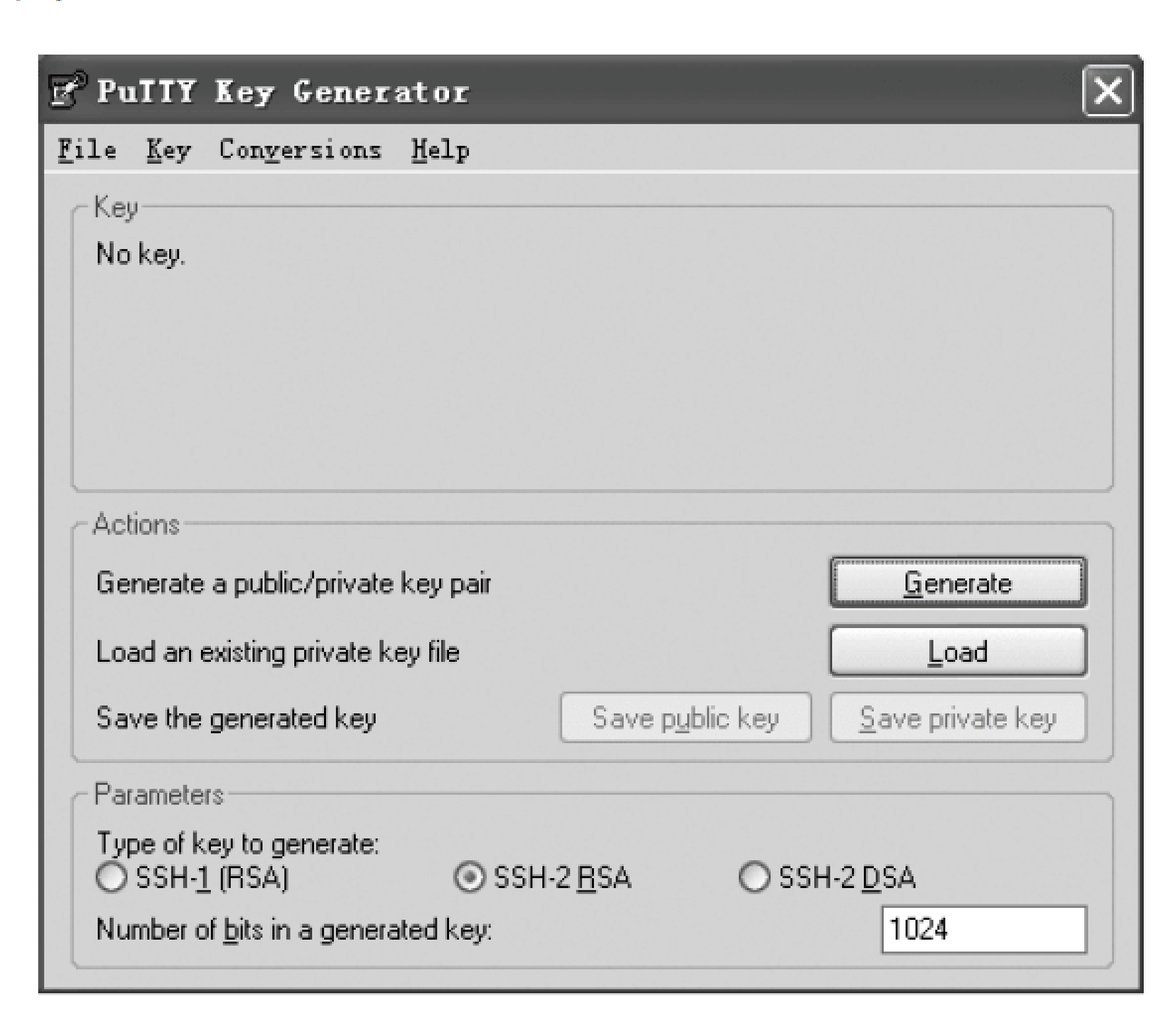


图 24.5　PuTTY gen 的主界面

在生成密钥的过程中，为了生成一些随机的数据，应在程序的窗口内随意移动鼠标（否则程序进度条不会改变）。密钥生成后，出于安全性的考虑，程序会提示输入保护私钥的口令短语“Key passphrase”，如图 24.6 所示。

3. 保存密钥

口令短语用于保护私钥，如果入侵者窃取了私钥，但没有口令短语，也不能使用该私钥。如果不想使用口令短语保护私钥，只需将该项留空即可。最后分别单击“Save public

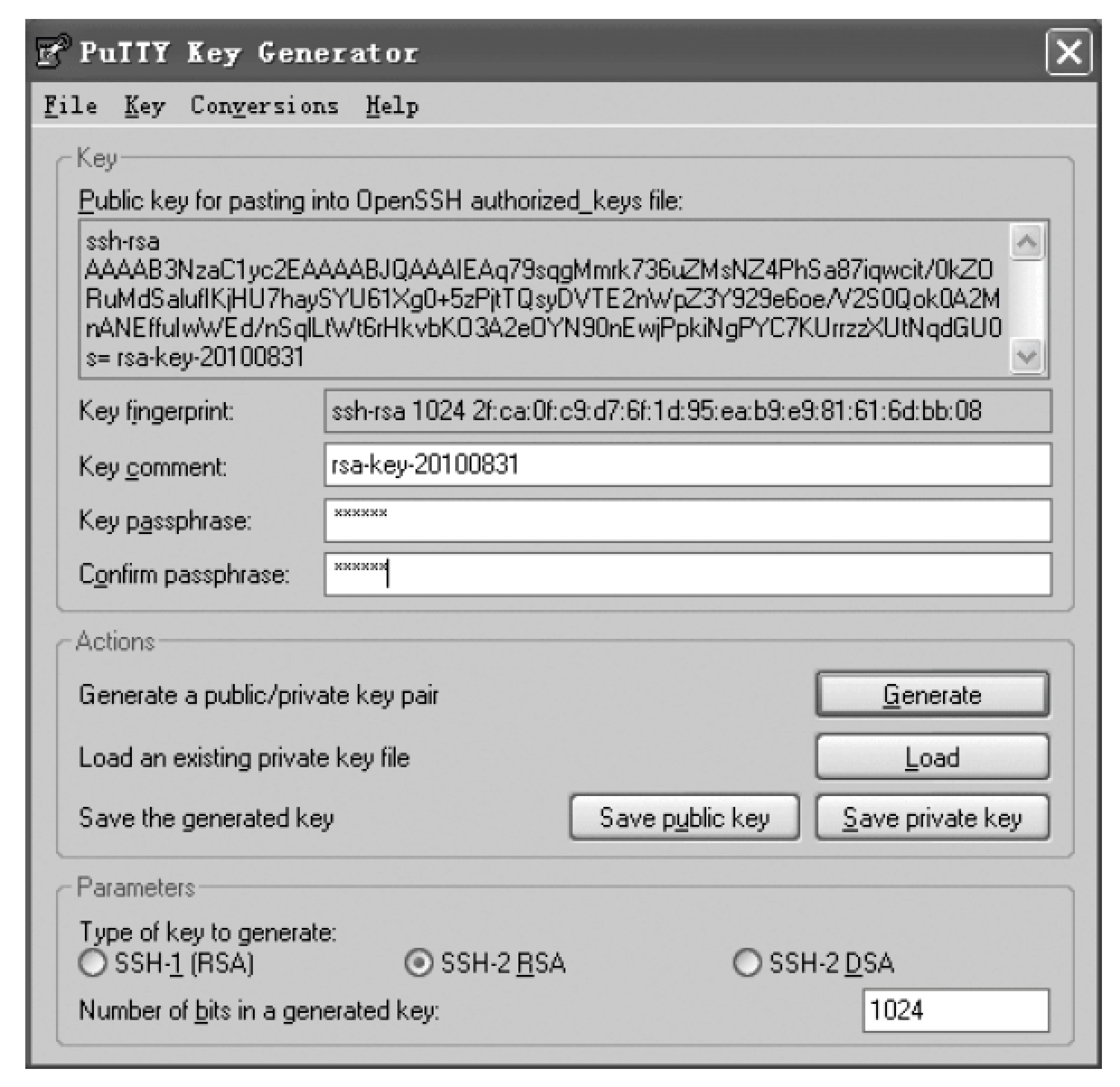


图 24.6　为私钥设置保护口令短语

key”和“Save private key”按钮将公钥和私钥保存成文件。本例保存公钥的文件名为xinya.pub，保存私钥的文件名为 xinya.ppk。

4. 传输公钥文件到 SSH 服务器

为了让 SSH 服务器能读取公钥文件，还要将文件传输到 SSH 服务器的用户主目录下的.ssh 子目录中（如果没有.ssh 目录，可手动建立）。因为公钥文件可以公开给所有用户，传输公钥文件时不必考虑安全问题，可以使用 FTP、电子邮件或复制等方法。

5. 轮换公钥文件格式

由于 PuTTYgen 产生的公钥文件格式与 OpenSSH 程序使用的格式不兼容，因此还要在 Linux 中使用 OpenSSH 软件包自带的 ssh-keygen 程序对其进行转换，输入如下命令：

```
ssh-keygen  -i  -f  /root/.ssh/xinya.pub >/root/.ssh/authorized_keys
```

在本例中，ssh-keygen 程序会将 PuTTYgen 产生的公钥文件 xinya.pub 转换成 OpenSSH 程序使用的格式，并将输出结果添加到/root/.ssh/authorized_keys 文件中。如果需要添加多个公钥文件，可以使用追加数据的方式添加到 authorized_keys 文件的末尾。例如，用下面的命令将公钥文件 testkey.pub 添加到 authorized_keys 文件的末尾：

```
ssh-keygen  -i  -f  /root/.ssh/testkey.pub >> /root/.ssh/authorized_keys
```

6. 连接 SSH 服务器

（1）运行 PuTTY 程序，在“Host Name”下的文本框中输入服务器的 IP 地址或域名（如 192.168.1.110），如图 24.7 所示。

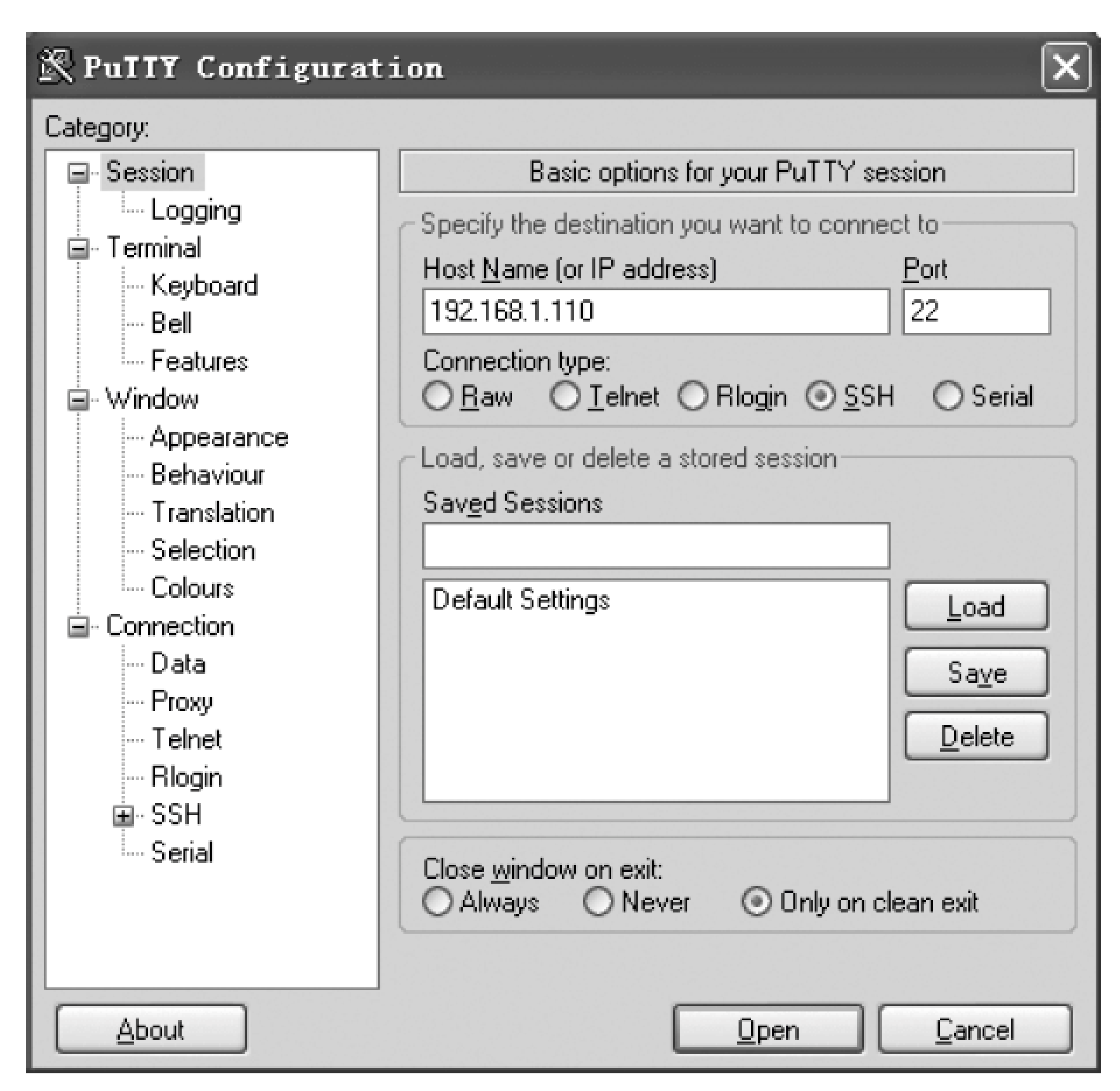


图 24.7　连接 SSH 服务器

（2）选择对话框左边的 Category 窗口的 Connection→SSH→Auth。

（3）在“Private key file for authentication：”下的文本框中输入私钥文件的路径，然后单击 Open 按钮连接，如图 24.8 所示。

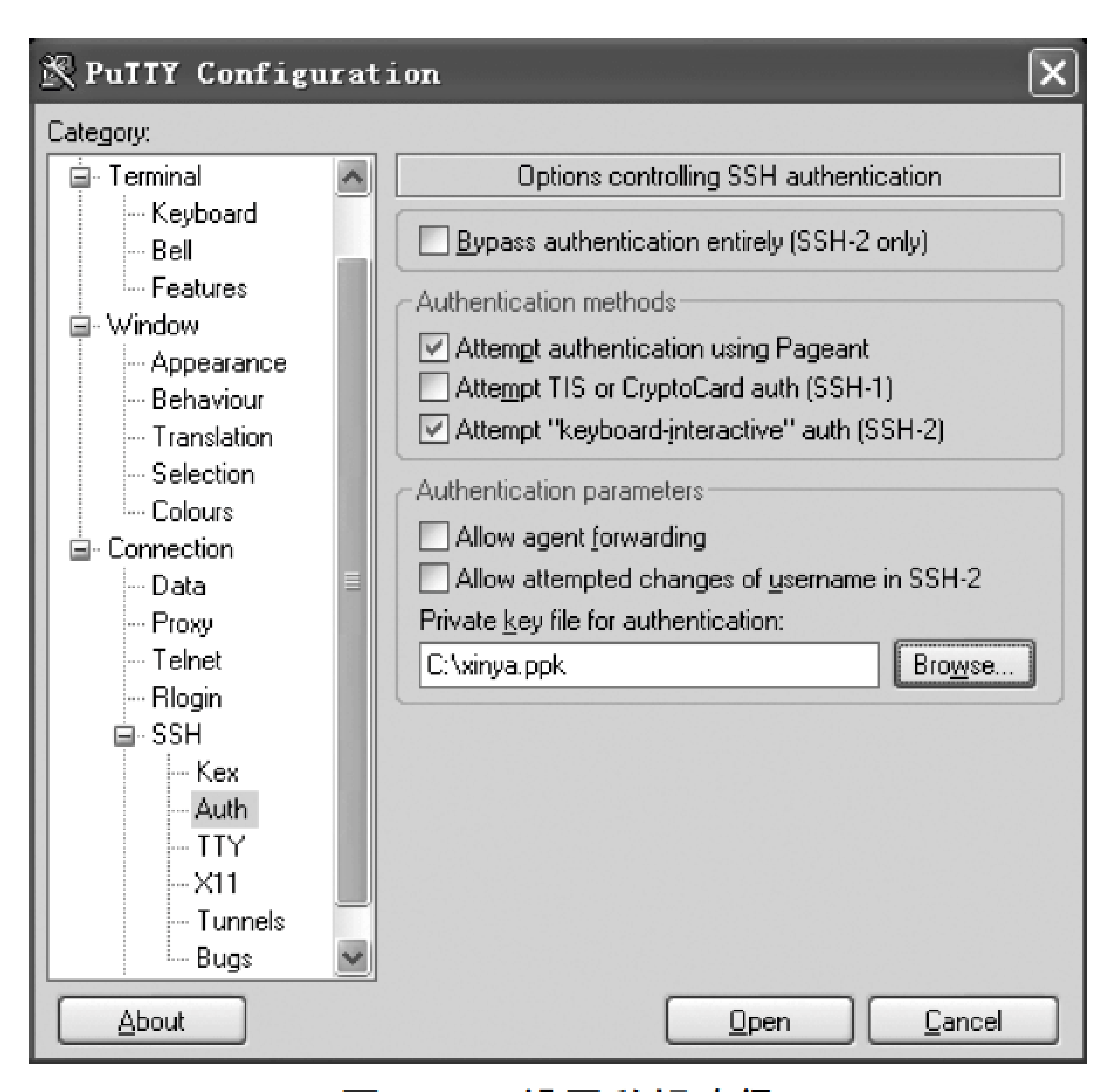


图 24.8　设置私钥路径

（4）PuTTY 成功地连接到 SSH 服务器后，服务器会提示输入登录用户名，如果使用了保护私钥的口令短语，则还会提示输入口令短语。在登录过程中不需要输入用户的密码，如图 24.9 所示。

```
root@xinya:~
login as: root
Authenticating with public key "rsa-key-20100831"
Passphrase for key "rsa-key-20100831":
Last login: Mon Aug 23 22:24:55 2010 from 192.168.1.110
[root@xinya ~]# whoami
root
[root@xinya ~]#
```

图 24.9　登录界面

24.6.5　在 openssh-clients 程序使用公钥认证

1．产生密钥

可以使用 openssh 软件包自带的 ssh-keygen 程序产生密钥，可使用“ssh-keygen　-t rsa”命令来产生，ssh-keygen 程序会提示输入保存密钥的路径和保护私钥的口令短语，默认密钥保存在当前用户的主目录下的.ssh 子目录中，私钥文件名为 id_rsa，公钥文件名为 id_rsa.pub。

```
[root@dns ~]# ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
84:cb:59:05:32:82:02:29:8e:18:99:ff:4b:d8:fa:3c root@dns.xinya.com
```

2．传输公钥文件到 SSH 服务器

为了让 SSH 服务器能读取公钥文件，还要将产生的文件 id_rsa.pub 传输到 SSH 服务器的用户目录下的.ssh 子目录中（如果没有.ssh 子目录，可手动建立），并改名为 authorized_keys。

3．连接 SSH 服务器

现在可以直接使用 ssh 命令登录到 SSH 服务器，操作如下：

ssh　服务器的 IP 地址或域名

如果 SSH 客户程序成功地连接到 SSH 服务器，程序会提示输入登录用户名；如果使用了保护私钥的口令短语，则还会提示输入口令短语。在登录过程中，不需要输入用户的密码。

参 考 文 献

1. 位元文化. Linux 管理与应用基础. 北京：清华大学出版社，2002.
2. 鸟哥. 鸟哥的 Linux 私房菜. 3 版. 王世江，改编. 北京：人民邮电出版社，2010.
3. Richard Petersen. Linux The Complete Reference. 4th ed. McGraw-Hill，2001.
4. 林惠琛，刘殊，尤国军. Red Hat Linux 服务器配置与应用. 北京：人民邮电出版社，2006.